박 사 학 위 논 문
Ph.D. Dissertation

음악 검색을 위한 유사도 기반 심층 학습

Similarity-based Deep Learning for Music Retrieval

2021

이 종 필 (李 鍾 筆 Lee, Jongpil)

한 국 과 학 기 술 원

Korea Advanced Institute of Science and Technology

박 사 학 위 논 문

음악 검색을 위한 유사도 기반 심층 학습

2021

이 종 필

한 국 과 학 기 술 원

문화기술대학원

음악 검색을 위한 유사도 기반 심층 학습

이 종 필

위 논문은 한국과학기술원 박사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2020년 12월 15일

심사위원장	남 주 한	(인) [illegible]
심 사 위 원	이 원 재	(인) [illegible]
심 사 위 원	이 경 면	(인) [illegible]
심 사 위 원	이 교 구	(인) [illegible]
심 사 위 원	Nicholas J. Bryan	(인) Nicholas J. Bryan

Similarity-based Deep Learning for Music Retrieval

Jongpil Lee

Advisor: Juhan Nam

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Culture Technology

Daejeon, Korea
December 22, 2020

Approved by

[illegible]

Juhan Nam
Professor of Graduate School of Culture Technology

The study was conducted in accordance with Code of Research Ethics[1].

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my thesis contains honest conclusions based on my own careful research under the guidance of my advisor.

DGCT 20175508

Jongpil Lee. Similarity-based Deep Learning for Music Retrieval. Graduate School of Culture Technology . 2021. 113+vii pages. Advisor: Juhan Nam. (Text in English)
이종필. 음악 검색을 위한 유사도 기반 심층 학습. 문화기술대학원 . 2021년. 113+vii 쪽. 지도교수: 남주한. (영문 논문)

Abstract

The number of music recordings that users can access is increasing, as composition and distribution of music becomes convenient through digitalization. In addition, user-generated music contents are distributed, and the functional use of music to amplify the mood or atmosphere of places (e.g. cafe, restaurant) or media content (e.g. video) is also increasing. Therefore, the act of searching for music contents is becoming very important. Most of the existing music search systems are mainly based on a catalogue-based search or a metadata-based search. Metadata-based search is to search for the exact matching song if you already know the song information such as song name or artist name, and catalogue-based search is to search for songs within categories such as genre, or mood, making detailed search difficult. If there is a way to find similar sounds-like songs to a query song, we will be able to browse and search many music recordings. Traditionally, the related technology is a recommendation algorithm based on the user's listening history. But basically, this recommendation algorithm cannot recommend songs that the user has not already consumed, and the recommendation is a passive act, and its use for active music search is limited. Therefore, the goal of this thesis is to explore content-based music search system that directly analyzes audio and searches through it.

In this dissertation, I explore the content-based music search methodology in three main aspects: a module for analyzing audio, a similarity-based deep learning method using various similarity concepts, and a methodology that opens up the possibility of new music search applications. To do that, the background methodology is explained in Chapter 2, an effective audio model is explored in Chapter 3, learning methods using various music similarity concepts are explored in Chapter 4, and a unified framework for similarity-based learning system that enables new music search application is explored in Chapter 5. More concretely, in chapter 3, I propose a more effective audio model that can directly perform on waveform instead of using spectrogram-based features. In chapter 4, I propose a similarity-based learning method that utilizes various music metadata with different levels of similarity concept. In chapter 5, I propose two new music search applications which are query-by-attribute and query-by-prototype by adding several techniques to the similarity-based deep learning methods. Through the exploration of this thesis, I hope we will be able to develop a better content-based music search system.

Keywords Music similarity, music search, content-based music retrieval, waveform-based audio model, objective metadata, multidimensional music similarity, classification, metric learning, disentangled representation

초 록

디지털화와 함께 음악의 작편곡 및 유통이 편리해짐으로써, 이용자들이 접근할 수 있는 음악 레코딩의 수가 증대되고 있다. 또한, 유저가 직접 제작하는 음악 컨텐츠들이 유통되기도 하며, 카페, 레스토랑, 영상컨텐츠의 무드나 분위기를 증폭시키기 위환 기능적 음악의 활용 또한 증대되고 있다. 이에, 원하는 음악을 찾는 행위는 매우 중요해지고 있다. 기존의 음악 검색은 대부분 곡의 제목 또는 아티스트 이름을 통한 검색 혹은 장르 등을 통한 카탈로그 기반 검색 방법이 주를 이룬다. 앞의 검색 방법은 이미 곡 정보를 알고 있는 경우에 정확한 그 곡을 검색하는 행위이고, 카탈로그 기반 검색은 세세한 검색이 힘들다는 단점이 있다. 특정 곡을 정하고 이와 비슷한 소리를 가지는 유사한 곡들을 검색하는 방법이 있다면 우리는 많은 음악 레코딩을 브라우징하며 검색할 수 있을 것이다. 전통적으로 사용자들의 음악 청취 이력을 기반으로 하는 추천 알고리즘이 연관된 기술이다. 하지만, 이러한 추천 알고리즘은 기본적으로 사용자가 이미 소비한 음원 컨텐츠에 대하여 추천이 가능하며, 추천 시스템은 수동적인 행위로서, 음원의 능동적 검색에는 그 사용예가 한정적이다. 이에, 오디오를 직접 분석하고 이를 통해 검색을 수행하는 컨텐츠 기반 음악 검색 시스템을 탐색하는 것이 본 학위의 목표이다.

본 학위논문에서는 크게 세 가지 측면에서 컨텐츠 기반 음악 검색 방법론을

탐구한다. 컨텐츠 기반 음악 검색 시스템에는 크게 오디오를 분석하는 모듈, 다양한 유사도의 유사도 기반 딥러닝 모델의 학습 방법, 그리고 다양한 어플리케이션을 위한 학습 및 활용 방법이 있다. 이에 본 학위논문에서는 Chapter 2에서 배경 방법론을 설명하고, Chapter 3에서 효과적인 오디오 모델에 대한 탐구, Chapter 4에서 다양한 음악 유사도 정의의 활용 방법, 그리고 Chapter 5에서 더 많은 어플리케이션을 위한 학습 방법론 및 활용 방법을 탐구한다. 구체적으로, Chapter 3에서는 기존의 스펙트로그램 기반 방법론을 파형 기반 방법론으로 수행함으로써 더욱 효과적이고 효율적인 오디오모델을 제안하였다. Chapter 4에서는 다른 유사도 수준을 가지는 다양한 음악 메타데이터 정보를 이용한 음악 유사도 모델의 학습방법을 제안하며, 이를 통해 다른 유사도 정의로 학습된 모델에 따른 모델의 학습 특성을 볼 수 있다. Chapter 5에서는 구성요소별 검색을 가능하게 하는 방법론 및 프로토타입을 이용한 검색 활용 방법을 탐험하여, 더 많은 컨텐트 기반 음악 검색 어플리케이션을 가능하게 하였다. 이러한, 본 학위 과정의 탐구를 통하여 우리는 더 나은 컨텐트 기반 음악 검색 시스템을 개발할 수 있게 된다.

핵심낱말 음악 유사도, 음악 검색, 컨텐츠 기반 음악 검색, 파형 기반 오디오 모델, 객관적인 메타데이터, 다차원 음악 유사도, 분류, 메트릭 학습, 풀린 차원

Contents

List of Tables

List of Figures

Chapter 1. Introduction

The music consumption experience has changed dramatically over the past decade. When vinyls were sold in the past, we thought carefully to buy one record at a record store. Since the release of CDs, we have been able to enjoy better sounding music at an affordable price. And, by the time when music began to be consumed in digital format (e.g. mp3), we were able to download music, and create our own playlists to listen to. In recent years, as music consumption has recently become a form of rental rather than a form of ownership such as streaming, music gradually has become a commodity that is produced and consumed more rapidly. Also, music is expanding to music video, live concerts, cover song, and remix. As the content creator base expands, user-created music is uploaded and distributed to services such as SoundCloud, YouTube, and Beatport.

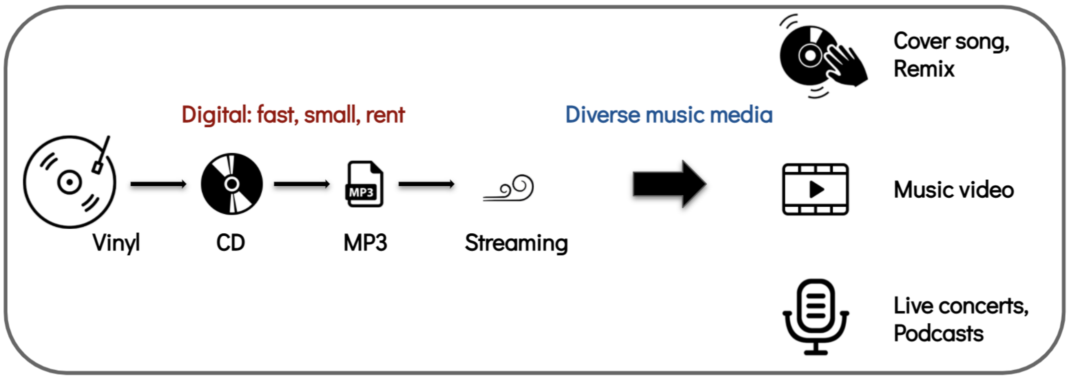


Figure 1.1: Music consumption trends.

Music consumption as a function is also expanding. Traditionally, music is used functionally in movies and broadcasts to boost mood or atmosphere, or to evoke memories of specific people and spaces. Recently, due to the expansion of the video contents, music is actively used functionally for background music in podcasts and YouTube videos. Accordingly, services such as Epidemic Sound, Artlist, and PremiumBeat that sell music in alternative licenses are increasing. Furthermore, music is used to create the mood of a space where multiple people stay together, such as a cafe, restaurant, and car environment, and this kind of use is becoming an important independent music consumption scenario. In addition, in A&R at a record company, a lot of time is spent on finding suitable artists, which will also be an important form of music consumption.

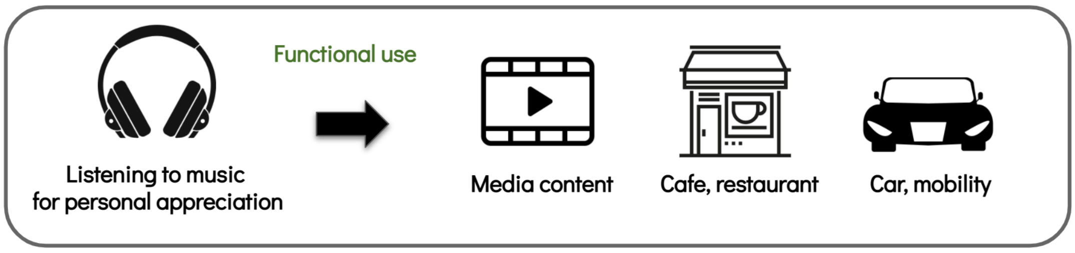
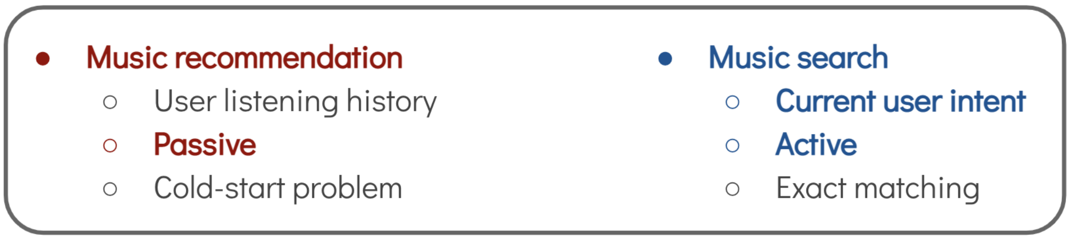


Figure 1.2: Functional use of music.

In this continuously changing music consumption experience, it is a very important problem for users to find various versions of music. Previously, this is mainly done by various forms of recommendation technology. A recommendation technology that has been successfully used in music streaming or video streaming services is to recommend music that users expect to like from their past content consumption history without instant queries from users. However, there are several problems in applying this recommendation algorithm to various contemporary music consumption behaviors. First, when various versions of music are newly created and user data is hard to accumulate, music recommendation is difficult and this is called a cold-start problem. Second, if there is a specific purpose of use such as functional music consumption, it is necessary to know the intention of what the user is looking for at that moment rather than the user's past history. Therefore, music search remains a unique and important problem apart from music recommendation.



- Music recommendation
    - User listening history
    - Passive
    - Cold-start problem
- Music search
    - Current user intent
    - Active
    - Exact matching

Figure 1.3: Music recommendation vs music search.

Current music search methods mainly consists of a basic search function such as metadata-based search in which the corresponding songs are searched when the artist and song name are known, and a catalogue-based search that finds music according to their categories such as genre or mood. However, metadata-based search finds mostly the corresponding song, so the scope of its specificity is very limited. Also, category-based search shows all songs in dozens of categories, making it difficult to select detailed songs because there are many songs in the category. If there exist a linear space in which songs with similar overall feel or mood are located closer and other songs are far away by directly analyzing musical characteristics, we will be able to have more powerful and useful user experiences.

Metadata-based search
Too specific
Catalogue-based search
Too broad

Figure 1.4: Traditional music search.

The field of content-based music retrieval makes more advanced search possible. In this case, music retrieval is possible by analyzing only audio content without metadata or user listening data. The basic concept is to project music audio into an organized space where musical characteristics are well represented, and to perform music search in this space. The process is as follows. Assuming that a well-built audio feature space exists, all music audio passes through this function to extract embedding features, and search for similar songs in the space. Also, if tag or artist-level entities can be mapped in this space, then query-by-example, query-by-tag, and query-by-artist will all be possible. We will explain these in more detail in the following section.

1.1 Content-based Music Retrieval

In the field of content-based music search, there are several query-by-example tasks [2–4]. Tasks such as audio identification or version identification are also included in the content-based music retrieval, but in this thesis, among these various

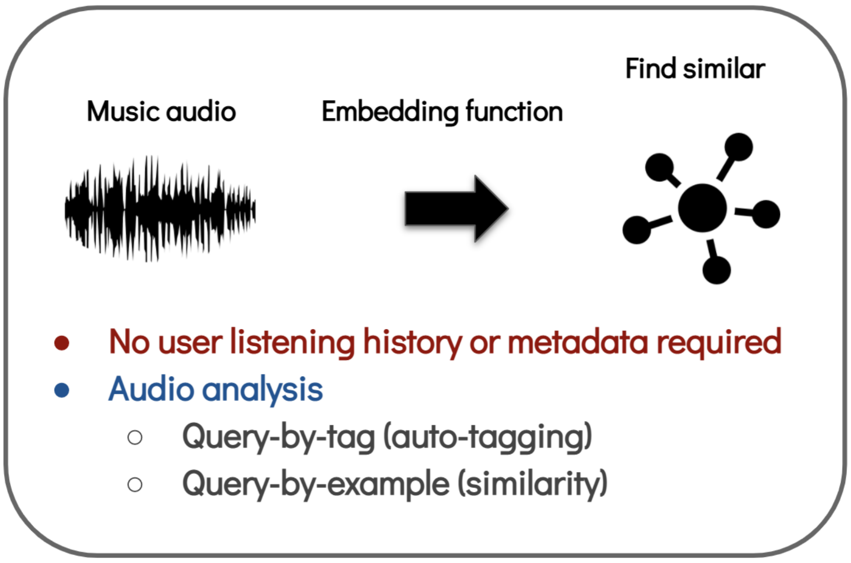

Music audio
Embedding function
Find similar
No user listening history or metadata required
Audio analysis
Query-by-tag (auto-tagging)
Query-by-example (similarity)

Figure 1.5: Content-based music retrieval.

content-based retrieval fields, our main target is not to find the exact song as the query song, but to find other music with similar overall feel or mood. Also, we focus on a song-level (or document-level) retrieval, which is a comparison between songs rather than between music segments. To do this, it is important to create an embedding function that demonstrate high-level musical characteristics, and it is important to consider the semantic gap and specificity aspects when creating this embedding function. These will be explained in the next sections.

1.1.1 Semantic Gap and Deep Embedding Function

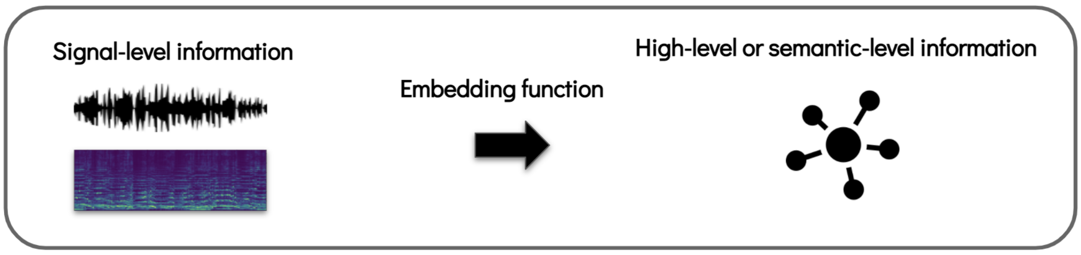


Figure 1.6: Semantic gap.

The semantic gap refers to the difference between high-level features that people generally use when searching for music and low-level features that can be extracted by computational methods from audio data. Here, the high-level feature refers to music descriptors level information, and the low-level feature refers to loudness, onset, chord, and tempo level information. It can be more human friendly if higher-level information can be used in search, but it is a difficult problem to build a system that extracts these high-level features. Recent advances in deep learning technology have narrowed this semantic gap in the area of image and audio. There are many hierarchical layers in different abstraction level from signal-level information to semantic-level information, deep learning models that have multiple nonlinear layers show good performance in modeling high-level information. In the image domain, in the past, low-level information such as color, texture, and shape was extracted and a predictive model was built on it. With the development of deep learning technology, it has recently evolved into an end-to-end model that analyzes the raw image itself. In the audio domain, low-level features such as MFCC and Chroma were used, and gradually developed into a spectrogram-based method, but there were no successful techniques for analyzing raw waveforms directly. Therefore, in this thesis, we explore a model that extracts high-level features directly from the raw waveforms.

1.1.2 Specificity and the Notion of Similarity

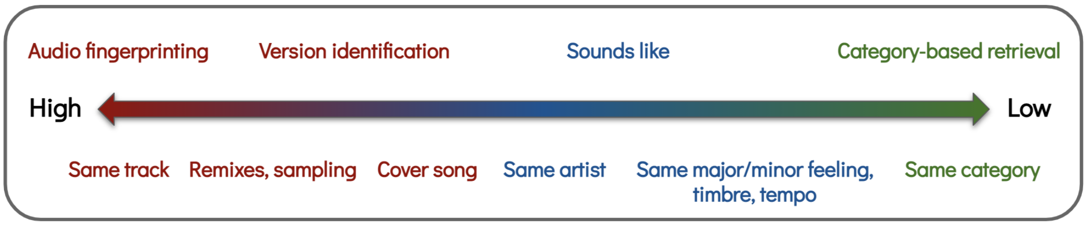


Figure 1.7: Specificity.

Specificity refers to how similar songs are compared to the query song when searching for music. For example, finding the same song as the query audio within a large database of content would have the highest specificity level, finding a style and genre similar to query audio would be low specificity, and sounds like query audio but not identical to the query audio would be in the middle. In this thesis, the main goal is to find a song that is most similar to a query song, but not identical to the query song, so that it can be used when searching for music. In order to create an similarity model in which these musical characteristics are preserved, various notions of similarity can be supervised in the model. For example, semantic tags that are labeled on music, acoustic features such as energy or tempo, and objective metadata such as artist, album, and track information may be utilized. Also, if there is an adjustable part in the embedding function so that we can control the specificity when searching, it will be of great help in user experience. Therefore, in this thesis, we explore the embedding function learning techniques using various notions of similarity, and develop a music search methodology capable of controlling specificity.

1.2 Related Research Topics

Research topics closely related to content-based music retrieval include audio-based music classification and tagging, and metric learning for music similarity. There is also an emerging field of disentangled multidimensional metric space. We will look at them in the next sections.

1.2.1 Music Classification and Tagging

The main purpose of music classification and tagging is to predict the tags that describe music from audio. In general, the tagging task is distinguished from the classification task in that tags are multi-labeled with words such as era, instrument, and style in addition to genre and mood. Therefore, in general, the tagging model is highly utilized, and many audio-based music classification and tagging model studies have been conducted based on the tagging task [5–8]. Thus, in the case of the predictive model verified in this task, it can be widely used in other tasks as a good deep embedding function that can bridge the semantic gap well. The last hidden layer of this deep embedding function is often used in transfer learning as an intermediate feature to other tasks. However, research on the use of this last hidden layer as a similarity space is insufficient. This thesis also aims to develop and utilize the efficient similarity space of the last hidden layer of the music classification and tagging model.

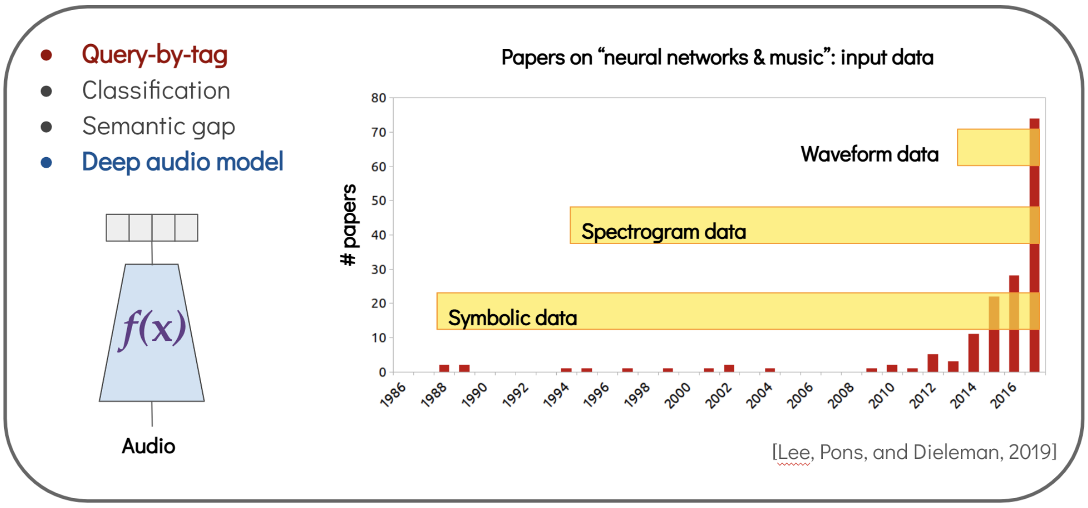


Figure 1.8: Music classification and tagging.

1.2.2 Metric Learning for Music Similarity

Audio-based music similarity has been mainly studied in the field of metric learning. Metric learning aims to create a metric space so that similar songs are distributed closer and other songs are distributed farther away. Various similarity notions and data types can be used for metric learning. For example, a classification type label annotated in music audio, and pair or triplet similarity data can be used for similarity learning. The concept of similarity can also be used in various ways [9, 10]. For example, information such as whether songs are from the same semantic label or not, whether they are songs of the same artist, or recommendation data can all be used. Prior studies were conducted with a linear metric learning method until 2010, but the performance was not satisfactory enough to fill the semantic gap and thus did not lead to a commercial level. At the present time, the performance of the audio model (or deep embedding function) itself has improved a lot with development of the deep learning, in this thesis, we explore metric learning for music similarity using a deep embedding function, and finds a way to use diverse music information at different levels of specificity.

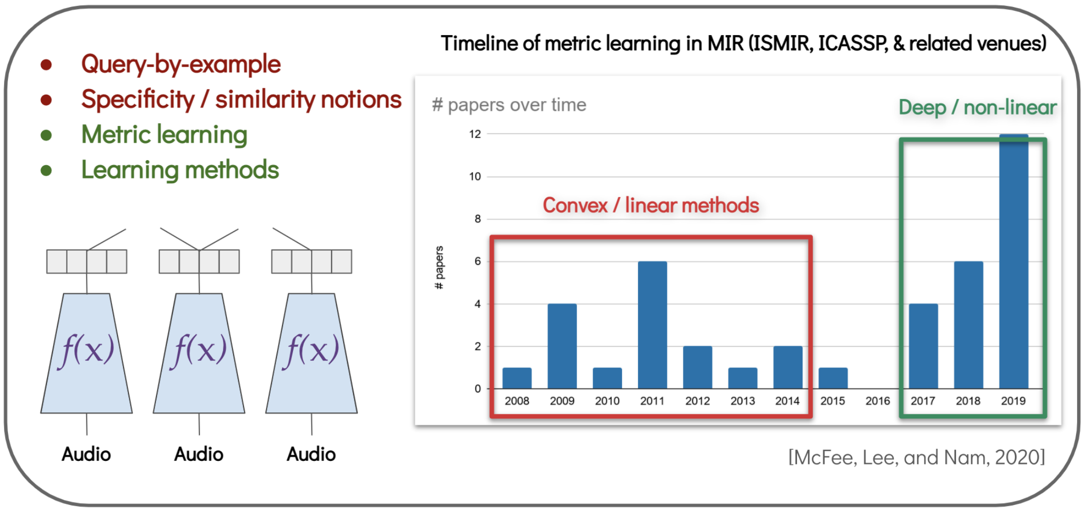


Figure 1.9: Metric learning for music similarity.

1.2.3 Disentangled Multidimensional Metric Space

Music is a multidimensional phenomenon. Songs have many dimensions of characteristics such as genre, tone, rhythm, and mood, and the two songs may be similar or different in some ways. If the similarity space could be separated by certain musical element characteristics, we can do interesting applications such as multidimensional music retrieval. For example, sometimes the user wants to search a playlist with a lighthearted mood, and at other times the user can search for music by era. To do that, in the image domain, Conditional Similarity Networks (CSN) has been proposed [11], and opens the possibility of multidimensional search by proposing a method of separating the search metric space using a masking function. Therefore, in this thesis, we apply CSN to the music representation learning, including classification and metric learning, and explore the possibility of multidimensional music search.

1.3 Research Contributions

The research contributions of this thesis can be largely summarized as follows. First, we explore the effective deep embedding function through the music classification and tagging task. Then, we explore how to make a similarity-based model using various similarity notions of music. Finally, a study was conducted on a method to enable multidimensional search by separating the metric space. Details of each contribution are as follows.

- First, in a situation where deep learning was applied to audio in the music classification and tagging task, but still intermediate features such as mel-spectrogram were mainly used, a model that can be learned directly from the raw waveform is explored, and as a result, it showed better performance than the mel-spectrogram.
- At the time of transition from linear metric learning to deep metric learning, we revisit metric learning of music with deep models. In particular, the composition of metric space using various notions of music similarity, such as tag, artist, album, track information, was explored. As a result, we showed that free artist labels are as useful as tag labels.
- We applied Conditional Similarity Networks (CSN) to deep metric learning for music similarity and open the possibility of multidimensional music retrieval. Furthermore, we proposed disentangled version of classification model, and showed that this type of disentangled classification model with normalization technique is very effective.

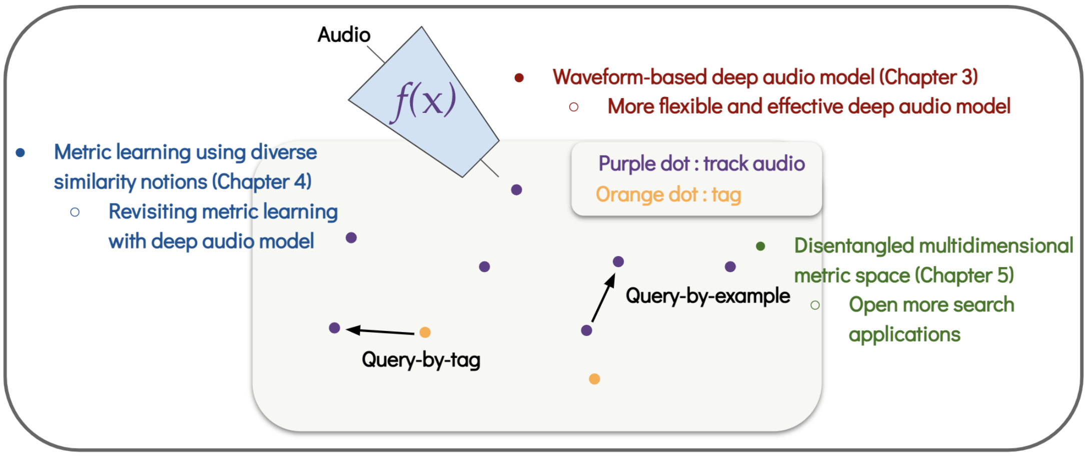

Audio
$f(x)$
Waveform-based deep audio model (Chapter 3)
More flexible and effective deep audio model
Metric learning using diverse similarity notions (Chapter 4)
Revisiting metric learning with deep audio model
Purple dot : track audio
Orange dot : tag
Disentangled multidimensional metric space (Chapter 5)
Open more search applications
Query-by-example
Query-by-tag

Figure 1.10: Contributions in geometric view.

Chapter 2. Background

The main modules for the development and use of similarity-based deep learning for music retrieval are deep embedding function, the notions of similarity, learning methods, and application and evaluation. Let's take a closer look at these.

2.1 Deep Embedding Function

First, deep embedding function, or deep audio model, basically refers to a deep learning model architecture that extracts high-level features from audio. From here, we define the deep embedding function as $f(x)$, x represents the audio example. In our main task, the song-level music retrieval, the training and application of the deep audio model should take into account the followings.

2.1.1 Music Audio and Convolutional Neural Networks

Music is a very complex audio, and I would like to compare this sound to the speech and scene sounds, which are other large fields of audio. I would like to say two dimensions: whether the sound sources are polyphonic with multiple sound sources, and whether the sound sources have a long-term structure. For example, speech is structured in sequence because it is mainly a single speaker and has a semantic meaning. On the other hand, the scene sound is polyphonic, but the individual sounds are not structured, they are occurred very naturally. Music is polyphonic, and the sound sources are structured and even these sounds are well mixed, making music a work of art. Therefore, it is important to understand the characteristics of these music audio in order to develop a good deep embedding function. Since sound is sequential data, we often think that sequence modeling algorithms such as Recurrent Neural Networks will work well, but many previous studies have not yet created a deep learning-based sequence modeling algorithm that works well for musical audio. This may be because both the areas related to this thesis, music classification and tagging, and metric learning for music similarity, all deal with a level close to high-level semantic search. For example, in comparing rock and funk, it may be more efficient to look at the composition of instruments and beats in a few seconds rather than the difference in the overall progression of the song. Therefore, the Convolutional Neural Network (CNN) is often used to model high-level features. CNN has a great advantage in extracting features related to local repetitive textures. This is possible because the CNN is made by overlapping filters of small sizes. As a learning method, an audio segment, usually between 1 and 10 seconds, is used to train the model and a song level labels are used as the label for the segments. The concept of multiple instance learning is used here. For example, some segments of audio may match with the song-level labels, and in the case where music is like K-pop having multiple genres within a song, some segments may be different from the song-level labels. However, by using these multiple instances (segment), we can give the model more degrees of freedom and make a model with better segment-level predictions. After training is complete, feed-forwarded features for all segments in the song are averaged and treated as a song-level feature.

2.1.2 Mel-Spectrogram vs Waveform

In these CNN models, the mel-spectrogram can be used as an input and the waveform can also be used as an input directly. Since the mel-spectrogram stacks the frequency-wise features on the vertical axis and the time-wise features on the frame, the dimension of a time is dramatically reduced compared to the waveforms. This makes the deep embedding function, a deep CNN model, more convenient to extract high-level features. However, since parameters such as the number of mel filters, window size, and hop size are fixed in mel-spectrogram, during the extraction of these frequency-based features, there are limitations in creating a deep embedding function that is more optimized for data and labels. Therefore, there have been attempts to take the waveform as an input directly [12]. Because waveforms are very large data in one dimension, the deep embedding function based on waveforms was initially developed in the form of simulating a mel-spectrogram. However, this approach does not show better performance than the mel-spectrogram, so the waveform-based approach is not widely used. The waveform-based approach has the advantage of being efficient because it does not require separate feature data storage. Therefore, it is important to develop a waveform-based approach that works well.

2.2 Different Notions of Similarity

When creating a similarity metric space, it is very important which similarity notion to use for training. If there exist pairs or triplets with similarity annotations, we can directly use them for training, but since these do not exist largely, we usually use song-level annotation data. With song-level annotations, songs in the same category can be treated as similar songs and songs in different categories can be regarded as different. There are three major song-level annotation data: semantic labels, objective metadata, and acoustic features, and we will look at them in detail.

2.2.1 Semantic Labels

Semantic labels are words such as genre, mood, style and theme that users annotated in music. It usually consists of a set of hundreds of words and can be a taxonomy created with great care, or a folksonomy refined from user's comments or reviews. Since all music is annotated with hundreds of words, we may think that there are too many songs in one category, and the specificity of the semantic labels is poor. However, since the music is multifaceted data, it is often multi-labeled, so it is possible to compare songs that share several words. So, semantic labels become the notion of similarity, which has a non-low specificity. However, these semantic labels belong to data that is time-consuming and expensive because people have to annotate them themselves.

2.2.2 Objective Metadata

Objective metadata refers to data that is automatically annotated with the release of an album or song. These include artist names, album names, and track names. This information is cheaper and more objective than semantic labels. For example, semantic labels may be judged differently depending on the propensity of annotators, but this kind of metadata is objective. On the other hand, these objective metadata is characterized by a very large number of class labels, but a relatively small number of instances or samples belonging to each class label. For example, the number of artists can reach tens of thousands, but the average number of songs released by individual artists can be only a few dozen.

2.2.3 Acoustic Features

The above semantic labels and objective metadata represent high-level information, and our goal is also to create a high-level feature space. However, when considering a music retrieval scenario, acoustic features such as tempo and loudness are low-level, but can be used very importantly when searching for music. For example, if we need a music that has a similar overall mood or feel to the query song, but only a bit faster or slowwer in tempo, then training the model with these kind of low-level features will be useful. This requires an algorithm to extract these acoustic features, which can be either a signal processing based algorithm or a deep learning based algorithm. The recent advances in deep learning technology have greatly improved the performance of music information retrieval algorithms, so even if it is a pseudo label extracted from algorithms rather than an exact label annotated by a person, it is quite useful.

2.3 Learning Methods

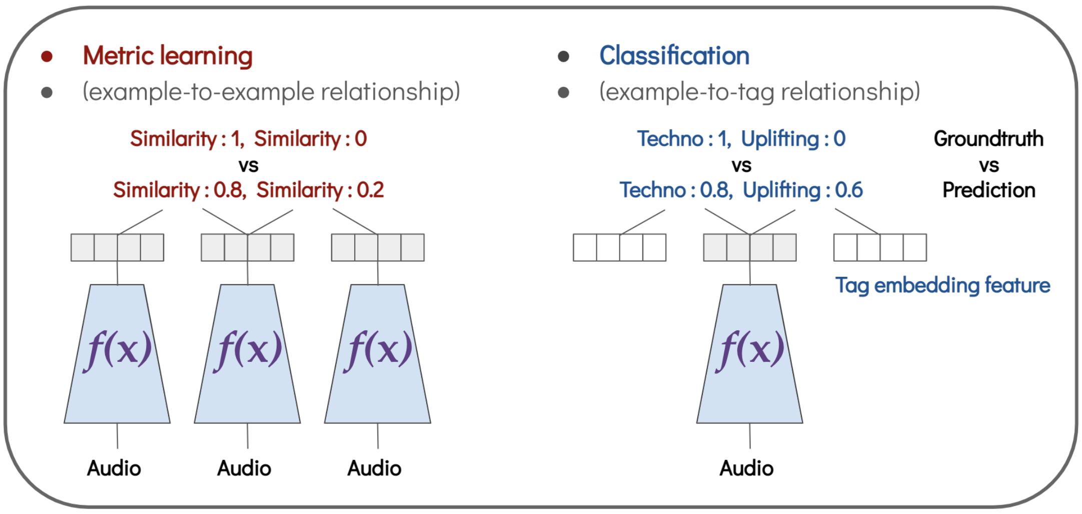

Metric learning
(example-to-example relationship)
Similarity : 1, Similarity : 0
vs
Similarity : 0.8, Similarity : 0.2
f(x)
Audio
Classification
(example-to-tag relationship)
Techno : 1, Uplifting : 0
vs
Techno : 0.8, Uplifting : 0.6
Groundtruth
vs
Prediction
Tag embedding feature

Figure 2.1: Metric learning vs classification.

There are mainly two ways to train deep embedding model: classification and metric training. In addition, there are metric space normalization and Conditional Similarity Networks (CSN) techniques that can be applied to these two learning methods. Classification is a method of learning a model using an example-to-tag relationship, and metric learning is a learning method using an example-to-example relationship that is created using some similairty notion. Metric space normalization enables better search by normalizing the search space, and CSN is a method that enables multidimensional search by explicitly dividing the metric space into several sub-dimensions. Let me explain them more in detail.

2.3.1 Classification

Classification is to predict an associated labels from music audio. The formula is as follows.

$$\hat{y} = Activation(W \cdot f(x) + b), \tag{2.1}$$

where $\hat{y}$ denotes prediction value, $\hat{y} \in \mathcal{R}^M$, and M is the number of class labels. $f(x)$ is a deep embedding function, $f(x) \in \mathcal{R}^N$, and N is the dimension of feature or metric space. W is a dimension of $M \times N$ matrix, and b is bias term. In addition, a combination of $CrossEntropyLoss$ and $Activation$ is used for model optimization. In general, a combination of softmax activation and categorical cross-entropy loss is used in single-label classification, and a combination of sigmoid activation and binary cross-entropy loss is used in multi-label classification. Here, if we ignore the activation and bias terms, then the formula becomes as follows.

$$\begin{gathered} W \cdot f(x) = [c_1, c_2, ..., c_m] \cdot f(x), \\ Metric(c_i, f(x)) = c_i \cdot f(x). \end{gathered} \tag{2.2}$$

In this equation, c_i is a feature vector of dimension m and plays a role as a class centroid for each label. Then, this class centroid forms a dot product based metric space with the deep embedding function $f(x)$. However, in the case of classification, the goal is not to build a well-established metric space, but to perform a label prediction well, so it is not suitable for searching in the metric space. This is because each class centroid c_i of W has a m dimension of freedom and forms a twisted projection with

$f(x)$.

2.3.2 Metric Learning

On the other hand, metric learning aims to build a metric space as priority.

$$Metric(f(x_i), f(x_j)), \tag{2.3}$$

as shown in this equation, in metric learning, it is easy to perform similarity search in metric space because it directly compares the output of deep embedding function of two examples. Here, the metric can be any distance or similarity metrics such as euclidean distance or cosine similarity. For convenience, I will explain this section with similarity based metric. The following triplet hinge loss is widely used as a loss for metric learning.

$$TripletLoss(x_a, x_p, x_n) = \max\{0, Metric(f(x_a), f(x_n)) - Metric(f(x_a), f(x_p)) + \Delta\}, \tag{2.4}$$

As inputs, three examples are taken: x_a, x_p, and x_n, each representing an anchor, positive and negative examples. Anchor and positive are the same class label or similar example, but not negative. In Equation 2.4, Δ means margin, and $TripletLoss$ is that when $Metric$ is based on similarity, the similarity value between anchor and positive is closer by margin than the similarity value between anchor and negative. Here, the triplet can be a triplet in which user's judgement is annotated, or it can also be a sampled triplet from class label annotations.

2.3.3 Metric Space Normalization

Since the metric space is used for similarity search, it is good to have a unit bounded metric space as it can provide a more compact structure of examples. Therefore, normalization technique is often used. In metric learning,

$$Metric(\frac{f(x_i)}{||f(x_i)||}, \frac{f(x_j)}{||f(x_j||}), \tag{2.5}$$

as above, a normalized metric is used and representative ones are normalized Euclidean distance or cosine similarity. Also, if this kind of normalization technique is applied in classification, then the Equation 2.2 becomes as follows.

$$Metric(c_i, \frac{f(x)}{||f(x)||}), \tag{2.6}$$

Here, now the deep embedding feature is unit bounded, so it is more useful for similarity search.

2.3.4 Conditional Similarity Networks

Furthermore, Conditional Similarity Networks (CSN) separates the metric space so that each sub-dimension of a metric space is responsible for each similarity notion [11]. This was first proposed in the image domain. This method uses higher hierarchy information in addition to the class label information as an additional supervision, such as genre, mood, and instrument. Then, the Equation 2.5 now becomes

$$Metric(\frac{f(x_i)}{||f(x_i)||} \circ m_s, \frac{f(x_j)}{||f(x_j||} \circ m_s), \tag{2.7}$$

where $\circ$ is Hadamard product and m_s is a mask. This mask is multiplied by the embedding feature and serves to construct a sub-metric space for each similarity notion. If this mask is also applied to the classification metric, it is as follows:

$$Metric(c_i \circ m_s, \frac{f(x)}{||f(x)||} \circ m_s). \tag{2.8}$$

If the model training is done with this masking function, then, we can search for music in each masked metric and it is called multidimensional search.

2.4 Application and Evaluation

This section introduces the various applications and evaluation methods of the deep metric model.

2.4.1 Tag-based Retrieval

First, tag-based retrieval is to search for related songs using music description words. In the case of the classification model, related songs can be searched by sorting the class prediction values of songs for each word in descending order. For metric

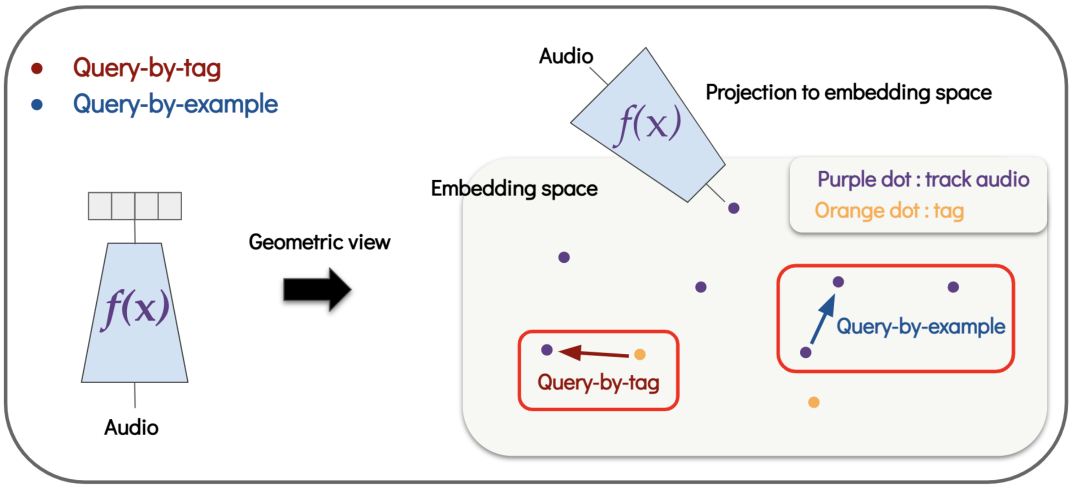


Figure 2.2: Applications in geometric view.

learning, we don't have these predicted values for each word, so we need to use different methods for tag prediction. One method is similar to the method used in prototypical network [13]. This method first averages the embedding features of all the songs that are annotated for each tag, and treats it as a class centroid. Then, the distance between this class centroid and the embedding feature of the song is measured and it is used as a tag prediction value. This can be expressed by the following equation.

$$c_i = \frac{1}{N} \sum_{x \in S_i} f(x), \tag{2.9}$$

where c_i is the i-th class centroid, S_i is the set of all songs annotated with the i-th class label, and N is the number of examples in a set S_i. This task is generally evaluated by measuring the ranking by tag and averaging the ranking score for all tags, and Area Under the ROC Curve (ROC-AUC) being the dominant evaluation metric.

2.4.2 Example-based Retrieval

Example-based retrieval is to perform a search between songs in a metric space when there is a deep embedding function. This query-by-example is simply done with the nearest neighbor search by calculating the distance between the embedding features of all songs. In the evaluation of this task, in the case of single-label, a simple k-nearest neighbor is performed to determine whether the nearest k examples belong to the same class as the query. However, in the case of multi-label annotations, evaluation is not easy, so we propose a multi-label Recall@K evaluation method in Chapter 5.

2.4.3 Multidimensional Retrieval

Multidimensional retrieval is to perform a search on each of these subspaces when the metric space is divided according to several similarity notions as shown in Section 2.3.4. When deep embedding function exists and the normalization technique is applied, the sub metric space according to the similarity notion s becomes as follows,

$$\frac{f(x)}{||f(x)||} \circ m_s. \tag{2.10}$$

This multidimensional retrieval can be evaluated as a task to predict separately sampled triplet according to each similarity notion.

2.4.4 Prototype-based Retrieval

Going further, we can perform a prototype-based retrieval. Prototype-based search is to search for similar songs by entity, such as artist and album names, rather than by music description words. This can be done by applying the method of creating class centroid that has been presented in Equation 2.9. For example, if the set S_i in Equation 2.9 is substituted with the set of all songs owned by a specific artist, not all songs belonging to one tag, then we can get an artist centroid or artist prototype. We can extend the application scenario by using these centroids. Applications such as query-by-artist or query-by-album can be examples.

2.4.5 User-rated Similarity Judgement

All of the evaluation methods listed above were quantitative measures of how well the model prediction was correct when there was a groundtruth label. For example, it is to measure whether a song searched as closest to the query song has the same groundtruth label as the query song. However, it may be different whether users actually feel how close the searched song is to the query song. Therefore, if we have similarity judgment data directly evaluated by the users, we will be able to better evaluate user's actual behavior. To do this, we create user-rated similarity judgment data in Chapter 5 of this thesis.

Chapter 3. SampleCNN: Sample-level Deep Convolutional Neural Networks Using Raw Waveforms

In this chatper, we will explore waveform-based deep audio model for music classification and tagging problem, then we further verify the model to more general audio classification problems such as speech commands classification and acoustic scene tagging tasks.

3.1 Music Classification and Tagging

Convolutional Neural Networks (CNN) have been applied to diverse machine learning tasks for different modalities of raw data in an end-to-end fashion. In the audio domain, a raw waveform-based approach has been explored to directly learn hierarchical characteristics of audio. However, the majority of previous studies have limited their model capacity by taking a frame-level structure similar to short-time Fourier transforms. We previously proposed a CNN architecture which learns representations using sample-level filters beyond typical frame-level input representations. The architecture showed comparable performance to the spectrogram-based CNN model in music auto-tagging. In this paper, we extend the previous work in three ways. First, considering the sample-level model requires much longer training time, we progressively downsample the input signals and examine how it affects the performance. Second, we extend the model using multi-level and multi-scale feature aggregation technique and subsequently conduct transfer learning for several music classification tasks. Finally, we visualize filters learned by the sample-level CNN in each layer to identify hierarchically learned features and show that they are sensitive to log-scaled frequency.

3.1.1 Problem

Convolutional Neural Networks (CNN) have been applied to diverse machine learning tasks. The benefit of using CNN is that the model can learn hierarchical levels of features from high-dimensional raw data. This end-to-end hierarchical learning has been mainly explored in the image domain since the break-through in image classification [14]. However, the approach has been recently attempted in other domains as well.

In the text domain, a language model is typically built in two steps, first by embedding words into low-dimensional vectors and then by learning a model on top of the word-level vectors. While the word-level embedding plays a vital role in language processing [15], it has limitations in that the embedding space is learned separately from the word-level model. To handle this problem, character-level language models that learn from the bottom-level raw data (e.g., alphabet characters) were proposed and showed that they can yield comparable results to the word-level learning models [16, 17].

In the audio domain, raw waveforms are typically converted to time-frequency representations that better capture patterns in complex sound sources. For example, spectrogram and more concise representations such as mel-filterbank are widely used. These spectral representations have served a similar role to the word embedding in the language model in that the mid-level representation are computed separately from the learning model and they are not particularly optimized for the target task. This issue has been addressed by taking raw waveforms directly as input in different audio tasks, for example, speech recognition [18–20], music classification [12, 21, 22] and acoustic scene classification [23, 24].

However, the majority of previous work have focused on replacing the frame-level time-frequency transforms with a convolutional layer, expecting that the layer can learn parameters comparable to the filter banks. The limitation of this approach was pointed out by Dieleman and Schrauwen [12]. They conducted an experiment of music classification using a simple CNN that takes raw waveforms or mel-spectrogram. Unexpectedly, their CNN models with the raw waveform as input did not produce better results than those with the spectral data as input. The authors attributed this unexpected outcome to three possible causes. First, their CNN models were too simple (e.g., a small number of layers and filters) to learn the complex structure of polyphonic music. Second, the end-to-end models need an appropriate non-linearity function that can replace the log-based amplitude compression in the spectrogram. Third, the first 1D convolutional layer takes raw waveforms in a frame-level which is typically several hundred samples long. The filters in the first 1D convolutional layer should learn all possible phase variations of periodic waveforms within the length. In spectrogram, the phase variation is removed.

We recently tackled the issues by stacking 1D convolutional layers using very small filters instead of a 1D convolutional layer with the frame-level filters, inspired by the VGG networks in image classification that is built with deep stack of 3×3 convolutional layers [25,26]. The sample-level CNN model has filters with very small granularity (e.g., 3 samples) in time for all convolutional layers. The results were comparable to those using mel-spectrogram in music auto-tagging. In this paper, we term the sample-level CNN architecture as SampleCNN and extend the previous work in three ways. First, we should note that SampleCNN takes four times longer training time than a comparable CNN model that takes mel-spectrogram. In order to reduce the training time, we progressively downsample the waveforms and report the effect on performance. By reducing the band-width of music audio this way, we will be able to find the cut-off frequency where the performance starts to become degraded. Second, we extended SampleCNN using multi-level and multi-scale feature aggregation [27]. The technique proved to be highly effective in music classification tasks. We additionally evaluate the extended model in transfer learning settings where the features extracted from SampleCNN can be used for three different datasets in music genre classification and music auto-tagging. We show that the proposed model achieves state-of-the-art results. Third, we visualize learned intermediate layers of SampleCNN to observe how the filters with small granularity process music signals in a hierarchical manner. In particular, we visualize them for each of sampling rates.

3.1.2 Related Work

There are a decent number of CNN models that take raw waveforms as input. The majority of them used large-sized filters in the first convolutional layer with various size of strides to capture frequency-selective responses which were carefully designed to handle their target problems. We termed this approach as frame-level raw waveform model because the filter and stride sizes of the first convolutional layer were chosen to be comparable to the window and the hop sizes of short-time Fourier transformation, respectively [12, 18–23].

There are a few work that used small filter and stride sizes in the first convolution layer (8 samples-sized filter [28] and 10 samples-sized filter [29, 30] at 16 kHz). However, the CNN models have only two or three convolution layers, which are not sufficient to learn the complex structure of the acoustic signals. In SampleCNN, we deepen the layers even more, thereby reducing the filter and stride sizes of the first convolution layer down to two or three samples.

3.1.3 Learning Models

Figure 3.1 illustrates three CNN models in music auto-tagging that we compare in our experiments. Note that they are actually general architectures and so can be applied to any audio classification tasks. In this section, we describe the three models in detail.

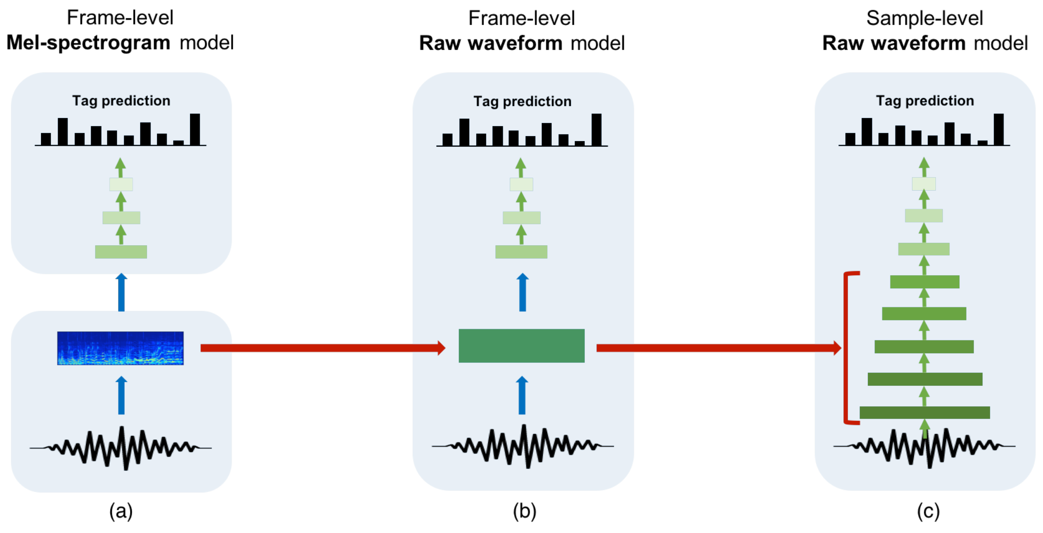


Figure 3.1: Comparison of (a) frame-level model using mel-spectrogram, (b) frame-level model using raw waveforms and (c) sample-level model using raw waveforms.

Frame-Level Mel-Spectrogram Model

This is the most common CNN model used in music classification. The time-frequency representation is usually regarded as either two-dimensional images [7, 31] or one-dimensional sequence of vectors [8, 12]. We only used one-dimensional(1D) CNN model for experimental comparisons because the performance gap between 1D and 2D models is not significant and the 1D model is directly comparable to models using raw waveforms.

Frame-Level Raw Waveform Model

In this model, a strided convolution layer is added beneath the bottom layer of the frame-level mel-spectrogram model. The strided convolution layer is expected to learn a filter-bank that returns a time-frequency representation. In this model, once the first strided convolution layer slides over the raw waveforms, the output feature map has the same dimensions as the mel-spectrogram. This is because the stride size, filter size, and the number of filters in the first convolution layer correspond to the hop size, window size, and the number of mel-bands in the mel-spectrogram, respectively. This configuration was used for the music auto-tagging task in [12, 21] and thus we used it as a baseline model.

Sample-Level Raw Waveform Model: SampleCNN

As described in Section 3.1.1, the approach using raw waveforms should be able to address log-scale amplitude compression and phase-invariance. Simply adding a strided convolution layer is not sufficient to overcome the issues. To improve this, we add multiple layers beneath the frame-level such that the first convolution layer can handle much smaller size of samples. For example, if the stride of the first convolution layer is reduced from 729 (=3^6) to 243 (=3^5), 3-size convolution layer and max-pooling layer are added to keep the output dimensions in the subsequent convolution layers unchanged. If we repeatedly reduce the stride of the first convolution layer this way, six convolution layers (five pairs of 3-size convolution and max-pooling layer following one 3-size strided convolution layer) will be added (we assume that the temporal dimensionality reduction occurs only through max-pooling and striding while zero-padding is used in convolution to preserve the size).

We generalized the configuration as m^n-SampleCNN where m refers to the filter size (or the pooling size) of intermediate convolution layer modules and n refers to the number of the modules. The first convolutional layer is different from the intermediate convolutional layers in that the stride size is equal to the filter size. An example of m^n-SampleCNN is shown in Table 3.1 where m is 3 and n is 9. Note that the network is composed of convolution layers and max-pooling only, and so the input size is determined to be *stride size of the first convolutional layer* $\times m^n$. In Table 3.1, as the stride size of the first convolution layer is 3, the input size is set to be 59049 (=3 $\times$ 3^9).

3^9-SampleCNN Model			
59,049 Samples (2678 ms) as Input			
Layer	**Stride**	**Output**	**# of params**
conv 3-128	3	$19,683 \times 128$	512
conv 3-128	1	$19,683 \times 128$	$49,280$
maxpool 3	3	6561×128	
conv 3-128	1	6561×128	$49,280$
maxpool 3	3	2187×128	
conv 3-256	1	2187×256	$98,560$
maxpool 3	3	729×256	
conv 3-256	1	729×256	$196,864$
maxpool 3	3	243×256	
conv 3-256	1	243×256	$196,864$
maxpool 3	3	81×256	
conv 3-256	1	81×256	$196,864$
maxpool 3	3	27×256	
conv 3-256	1	27×256	$196,864$
maxpool 3	3	9×256	
conv 3-512	1	9×512	$393,728$
maxpool 3	3	3×512	
conv 3-512	1	3×512	$786,944$
maxpool 3	3	1×512	
conv 1-512	1	1×512	$262,656$
dropout 0.5	–	1×512	
sigmoid	–	50	25650
Total params			2.46×10^6

Table 3.1: SampleCNN configuration. In the first column (Layer), "conv 3-128" indicates that the filter size is 3 and the number of filters is 128.

3.1.4 Extension of SampleCNN

Multi-Level and Multi-Scale Feature Aggregation

Music classification tasks, particularly music auto-tagging among others, have a wide variety of labels in terms of genre, mood, instruments and other song characteristics. Especially, they are positioned in different hierarchical levels and time-scales. For example, some words related to instrument ones, such as guitar and saxophone, describe objective sound sources which are usually local and repetitive within a song, whereas other labels related to genre or mood, such as rock and happy, are dependent on a larger context of music and are more complicated. In order to address this issue, we recently proposed multi-level and multi-scale feature aggregation technique [27].

The technique is conducted by combining multiple CNN models. This assumes that the hidden layers of each CNN model represent different levels of features and the models with different input sizes provide even richer feature representations by capturing both local and global characteristics of the music. In [27], they showed that different level and time-scale features have different performance sensitivity to individual tags and thus combining them all together is the best strategy to improve performance. In this work, we replace the simple CNN architectures that take mel-spectrogram as input in [27] with SampleCNNs, taking different input sizes (e.g., 700 ms to 3.5 s). Once we train the SampleCNNs as supervised feature extractors, we slide each of them over a song clip (e.g., about 30 s) and obtain features from the last three hidden layers. We then summarize them by a combination of max-pooling and average-pooling. Finally, we concatenate the multi-level and multi-scale features and feed them to a simple neural networks with two fully-connected layers to make a final prediction.

Transfer Learning

The multi-level and multi-scale feature aggregation approach can be used in a transfer learning setting by using different datasets or target tasks for the final classification after training the SampleCNNs. Especially, when the target dataset size is comparably small to the model capacity, transferred parameters can yield better performance on the target task rather than parameters trained from the innate target dataset. The applicability of transfer learning using a frame-level raw waveform model has been explored in the speech domain [29]. Here, we examine it using the sample-level raw waveform model for music genre classification and music auto-tagging with different datasets.

3.1.5 Experimental Setup

Datasets

We validate the effectiveness of the proposed method on different sizes of datasets for music genre classification and auto-tagging. All dataset splits are available on the link [?]. The details of each dataset are as follows. The numbers in the parenthesis indicate the split of training, validation and test sets.

- GTZAN [32]: 930 songs (443/197/290) [1], genre classification (10 genres).
- MagnaTAgaTune (MTAT) [34]: 21,105 songs (15,244/1529/4332), auto-tagging (50 tags)
- Million Song Dataset with Tagtraum genre annotations (TAGTRAUM): 189,189 songs (141,372/10,000/37,817) [2], genre classification (15 genres)
- Million Song Dataset with Last.FM tag annotations (MSD) [36]: 241,889 songs (201,680/11,774/28,435), auto-tagging (50 tags)

We primarily examined the proposed model on MTAT and then verified the effectiveness of our model on MSD which is much larger than MTAT[3]. We filtered out the tags and used most frequently labeled 50 tags in both datasets, following the previous work [7, 12, 31]. Also, all songs in the two datasets were trimmed to 29.1 s long. For transfer learning experiments, the model is first trained with the largest dataset, MSD, and the pre-trained networks are transferred to other three datasets. The evaluation

[1]This is a fault-filtered split designed to avoid the repetition of artists across the training, validation and test sets [33].

[2]This is a stratified split with 80% training data of the CD2C version [35].

[3]MTAT contains 170 h long audio and MSD contains 1955 h long audio in total.

is conducted with area under receiver operating characteristic (AUC) for auto-tagging datasets and accuracy for genre classification datasets.

Training Details

We used sigmoid activation for the output layer and binary cross entropy loss as the objective function to optimize. For every convolution layer, we used batch normalization [37] and ReLU activation. We should note that, in our experiments, batch normalization plays a vital role in training the deep models that take raw waveforms. We applied dropout of 0.5 to the output of the last convolution layer and minimized the objective function using stochastic gradient descent with 0.9 Nesterov momentum. The learning rate was initially set to 0.01 and decreased by a factor of 5 when the validation loss did not decrease more than 3 epochs. A total decrease of 4 times, the learning rate of the last training was 0.000016. Also, we used batch size of 23 for MTAT and 50 for MSD, respectively.

Mel-Spectrogram and Raw Waveforms

In the mel-spectrogram experiments, window sizes of 3^6, 3^5 and 3^4 are used to match up to the filter sizes in the first convolution layer of the raw waveform model as shown in Table 3.2. FFT size was set to 729 (=3^6) in all experiments. When the window is less than the FFT size, we zero-padded the windowed frame. The linear frequency in the magnitude spectrum is mapped to 128 mel-bands and the magnitude compression is applied with a nonlinear curve, $\log(1+C|A|)$ where A is the magnitude and C is set to 10. Also, we conducted the input normalization simply by dividing the standard deviation after subtracting mean value of entire input data. On the other hand, we did not perform the input normalization for raw waveforms.

3^n Models, 59,049 Samples as Input	n	**Window Size (Filter Size)**	**Hop Size (Stride Size)**	**AUC**
Frame-level (mel-spectrogram)	4	729	729	0.9000
	5	729	243	0.9005
	5	243	243	0.9047
	6	243	81	0.9059
	6	81	81	0.9025
Frame-level (raw waveforms)	4	729	729	0.8655
	5	729	243	0.8742
	5	243	243	0.8823
	6	243	81	0.8906
	6	81	81	0.8936
Sample-level (raw waveforms)	7	27	27	0.9002
	8	9	9	0.9030
	9	3	3	0.9055

Table 3.2: Comparison of three CNN models with different window size (filter size) and hop size (stride size). n represents the number of intermediate convolution and max-pooling layer modules, thus 3^n times hop (stride) size of each model is equal to the number of input samples.

As described in Section 3.1.3, m refers to the filter size (which can be compared to a window size of FFT in the spectrogram) or pooling size (which also can be compared to a hop size of FFT in the spectrogram) of the intermediate convolution layer modules, and n refers to the number of the modules. In our previous work, we adjusted m from 2 to 5 and increased n according to the configuration of m^n-SampleCNN [25]. Among them, 3^9-SampleCNN model with 59049 samples as input worked best and thus we fix our baseline model to it. In this configuration, we can increase the filter size and stride size in the first layer by decreasing the layer depth to conduct comparison experiments between the frame-level models and the sample-level model. For example, if the hop size or the stride size of the first convolutional layer is 729 in either the frame-level mel-spectrogram model or the frame-level raw waveform model, 4 convolutional modules with 3-sized filters are added when the input size is 59,049 samples.

Downsampling

The downsampling experiments are performed using the MTAT dataset. 3^9-SampleCNN model is used with audio input sampled at 22,050 Hz. For other sampling rate experiments, we slightly modified the model configuration so that the models used for different sampling rate can have similar architecture and similar input seconds to those used in 22,050 Hz. In our previous work [25], we found that the filter size did not significantly affect performance once it reaches the sample-level (e.g., 2 to 5 samples), while the input size of the network and total layer depth are important. Thus, we configured the models as described in Table 3.3. For example, if the sampling rate is 2000 Hz, the first four modules use 3-sized filters and the rest 6 modules use 2-sized filters to make the total layer depth similar to the 3^9-SampleCNN. Also, 3-sized filters are used for the first four modules in all models for fairly visualizing learned filters.

Sampling Rate	**Input (in Milliseconds)**	**Models**	**# of Parameters**
2000 Hz	5184 samples (2592 ms)	3-3-3-3-2-2-2-2-2-2	1.80×10^6
4000 Hz	10,368 samples (2592 ms)	3-3-3-3-2-2-2-4-2-2	1.93×10^6
8000 Hz	20,736 samples (2592 ms)	3-3-3-3-2-2-4-4-2-2	2.06×10^6
12,000 Hz	31,104 samples (2592 ms)	3-3-3-3-3-2-4-4-2-2	2.13×10^6
16,000 Hz	43,740 samples (2733 ms)	3-3-3-3-3-3-3-5-2-2	2.19×10^6
20,000 Hz	52,488 samples (2624 ms)	3-3-3-3-3-3-3-3-4-2	2.32×10^6
22,050 Hz	59,049 samples (2678 ms)	3-3-3-3-3-3-3-3-3-3	2.46×10^6

Table 3.3: Models, input sizes, and number of parameters used in the downsampling experiment. In the third column (Models), each digit from left to right stands for the filter size (or the pooling size) of the convolutional module of SampleCNN from bottom to top. Thus, the number of digits represents the layer depth of each model.

Combining Multi-Level and Multi-Scale Features

For the multi-level and multi-scale experiments described in Table 3.4, we used total 8 models including 2^{13}, 2^{14}, 3^8, 3^9, 4^6, 4^7, 5^5 and 5^6-SampleCNNs. Also, two fully connected layers with 4096 neurons in each layer are used as classifier.

Features from SampleCNNs Last 3 Layers (Pre-trained with MTAT)	MTAT
3^9 model	0.9046
3^8 and 3^9 models	0.9061
2^{13}, 2^{14}, 3^8 and 3^9 models	0.9061
2^{13}, 2^{14}, 3^8, 3^9, 4^6, 4^7, 5^5 and 5^6 models	0.9064

Table 3.4: Comparison of various multi-scale feature combinations.

Transfer Learning

The source task for the transfer learning is fixed to music auto-tagging using MSD because the dataset contains the largest set of music. In this experiment, 3^9-SampleCNN was used. We examined the proposed model on three target datasets for genre classification and auto-tagging. We also examined the performance differences when using features from multiple levels of the pre-trained CNNs and also their combinations.

3.1.6 Results and Discussion

Mel-Spectrogram and Raw Waveforms

Table 3.2 shows that the sample-level raw waveform model achieves results comparable to the frame-level mel-spectrogram model. Specifically, we found that using a smaller hop size (81 samples $\approx$ 4 ms) worked better than those of conventional approaches (about 20 ms) in the frame-level mel-spectrogram model. However, if the hop size is less than 4 ms, the performance degraded. An interesting finding from the result of the frame-level raw waveform model is that when the filter length is larger than the stride, the accuracy is slightly lower than the models with the same filter length and stride. We interpret that this result is due to the learning ability of the phase variance. As the filter size decreases, the extent of phase variance that the filters should learn is reduced.

Effect of Downsampling

During the experiments, we observed that the training time of the proposed SampleCNN is about four times longer than the frame-level mel-spectrogram model because the proposed model has more network parameters with deeper layers. In order to reduce the training time, we downsampled the audio with a set of lower sampling rates including 2000, 4000, 8000, 12,000, 16,000, 20,000 Hz. This can be regarded as a time-domain counterpart of in linear-to-mel mapping in that both reduce the dimensionality of input and preserve low-frequency content. The results in Table 3.5 show that the performance is maintained down to 8000 Hz but it starts to be degraded from 4000 Hz. This may indicate that the relevant information to the task is concentrated below 4000Hz (the Nyquist frequency of 8000 Hz). Also, we report the training time ratio of the models taking re-sampled audio to the model using 22,050 Hz signal as input. At the expense of the accuracy, the training time can be reduced to about half.

Sampling Rate	**Training Time (ratio to 22050 Hz)**	**AUC**
2000 Hz	0.23	0.8700
4000 Hz	0.41	0.8838
8000 Hz	0.55	0.9031
12,000 Hz	0.69	0.9033
16,000 Hz	0.79	0.9033
20,000 Hz	0.86	0.9055
22,050 Hz	1.00	0.9055

Table 3.5: Effect of downsampling on the performance and training time. We matched the depth of the models taking different sampling rate to the 3^9-SampleCNN. For example, if the sampling rate is 2000 Hz, the first four convolutional modules use 3-sized filters and the rest 6 modules use 2-sized filters to make the total layer depth similar to the 3^9-SampleCNN.

Effect of Multi-Level and Multi-Scale Features

To measure the effect of multi-level and multi-scale feature combination, we experimented with several settings in Table 3.4. The SampleCNN models are first trained on MTAT dataset, then this pre-trained networks are used as feature extractors for the MTAT dataset again. The results show that as more features are fusioned, the performance increases. This can be viewed similar to an ensemble method, however our approach is distinguished from it in that the feature aggregation is performed on activations of the hidden layers, not on the prediction values.

Transfer Learning and Comparison to State-of-the-Arts

In Table 3.6, we show the performance of the SampleCNN model and the transfer learning experiments (the bottom four lines). The results achieved state-of-the-art results on three datasets except for MSD. However, when considering that the model used in [27] utilized both multi-level and multi-scale features, the AUC score (0.8842) obtained from multi-level features only seems to be reasonable. Also, we can see that the multi-level and multi-scale aggregation technique generally improves the performance, particularly in GTZAN.

MODEL	GTZAN (Acc.)	MTAT (AUC)	TAGTRUM (Acc.)	MSD (AUC)
Bag of multi-scaled features [38]	-	0.898	-	-
End-to-end [12]	-	0.8815	-	-
Transfer learning [39]	-	0.8800	-	-
Persistent CNN [40]	-	0.9013	-	-
Time-frequency CNN [41]	-	0.9007	-	-
Timbre CNN [42]	-	0.8930	-	-
2-D CNN [7]	-	0.8940	-	0.851
CRNN [31]	-	-	-	0.862
2-D CNN [33]	0.632	-	-	-
Temporal features [43]	0.659	-	-	-
CNN using artist-labels [44]	0.7821	0.8888	-	-
multi-level and multi-scale features (pre-trained with MSD) [27]	0.720	0.9021	0.766	0.8878
SampleCNN (3^9 model) [25]	-	0.9055	-	0.8812
-3 layer (pre-trained with MSD)	0.778	0.8988	0.760	0.8831
-2 layer (pre-trained with MSD)	0.811	0.8998	0.768	0.8838
-1 layer (pre-trained with MSD)	0.821	0.8976	0.768	0.8842
last 3 layers (pre-trained with MSD)	0.805	0.9018	0.768	0.8842

Table 3.6: Comparison with previous work. We report SampleCNN results on MagnaTAgaTune (MTAT) and Million Song Dataset (MSD). Furthermore, the result acquired from multi-level and multi-scale feature aggregation technique is also reported at the bottom 4 lines. "-n LAYER" indicates features of n layers below from the output are used for the transfer learning setting.

3.1.7 Visualization

In this section, we investigate two visualization techniques that can broaden our understanding of the learned hierarchical features in SampleCNN.

Learned Filters

Previous work in the music domain is limited to visualizing learned filters only on the first convolution layer [12, 21, 45] or visualizing responses after a filter is applied on a specific input [46, 47]. The gradient ascent method has been proposed for directly seeing what is learned at a filter [48] and this technique has provided deeper understanding of what convolutional neural networks learn from images [49, 50]. We applied the technique to our SampleCNN to observe how each filter in a layer processes the raw waveforms. The gradient ascent method is as follows. First, we generate random noise and back-propagate the errors in the network. The loss is set to the target filter activation. Then, we add the bottom gradients to the input with gradient normalization. By repeating this process several times, we can obtain the accumulated gradients-based waveform like signal at the input which is optimized to maximize the target filter activation. Examples of learned filters at each layer are in Figure 3.2. Although we can find the patterns that low-frequency filters are more visible along the layer, the estimated filters are still noisy. To show the patterns more clearly, we visualized them as spectrum in the frequency domain and sorted them by the frequency of the peak magnitude in Figure 3.3.

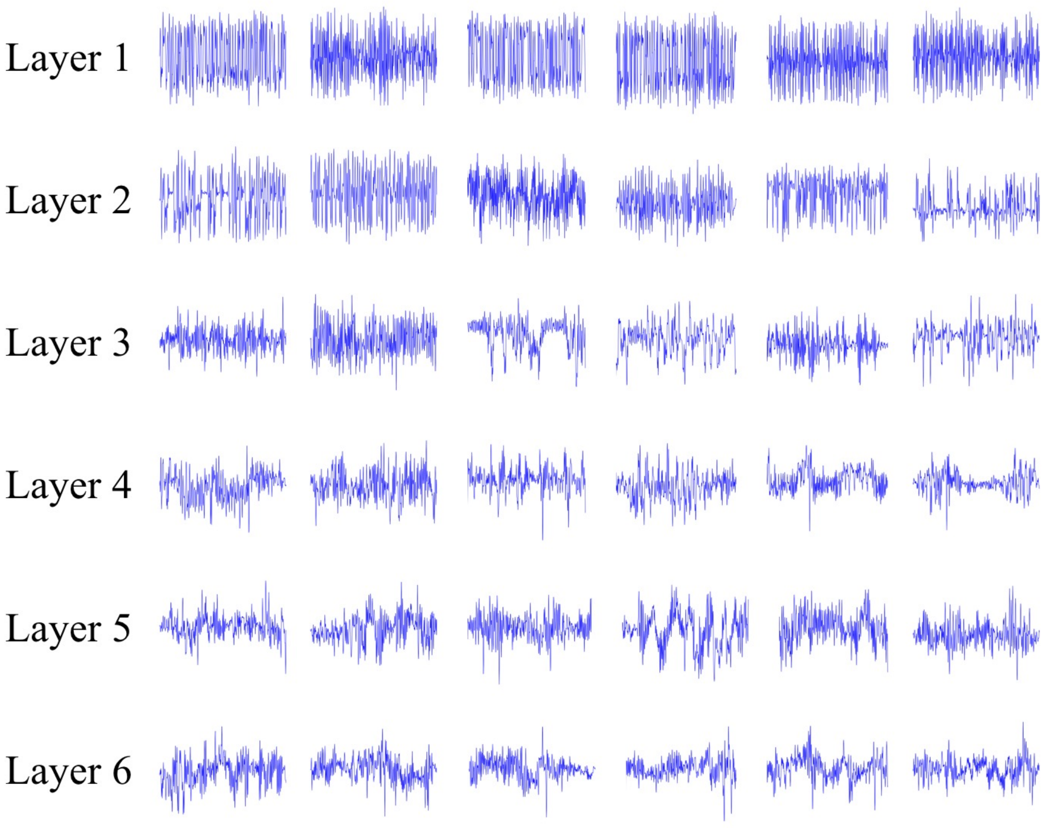


Figure 3.2: Examples of learned filters at each layer.

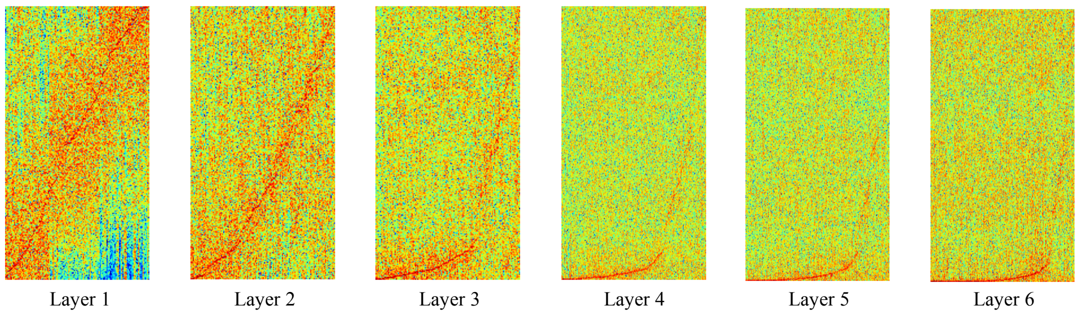


Figure 3.3: Spectrum of the estimated filters in the intermediate layers of SampleCNN which are sorted by the frequency of the peak magnitude. The x-axis represents the index of the filter, and the y-axis represents the frequency ranged from 0 to 11 kHz. The model used for visualization is 3^9-SampleCNN with 59,049 samples as input. Visualization was performed using the gradient ascent method to obtain the accumulated gradient-based input waveform like signal that maximizes the activation of a filter in the layers. To effectively find the filter characteristics, we set the input size to 729 samples which is close to a typical frame size.

Note that we set the input waveform estimate to 729 samples in length because, if we initialize and back-propagate to the whole input size of the networks, the estimated filters will have large dimensions such as 59,049 samples in computing spectrum. Thus, the results are equivalent to spectra from a typical frame size. The layer 1 shows the three distinctive filter bands which are possible with the filter size with 3 samples (say, a DFT size of 3). The center frequency of the filter banks increases linearly in low frequency filter banks but, as the layer goes up, it progressively becomes steeper in high frequency filter banks. This nonlinearity was found in learned filters with a frame-level end-to-end learning [12] and also in perceptual pitch scales such as mel or bark.

Finally, we visualized spectrum of the learned filter for each sampling rate up to 4th layers. In Figure 3.4, we can observe that all SampleCNN models focus (or zoom in) on the important low-frequency bands. We can also find that they show similar non-linear patterns to those in Figure 3.3.

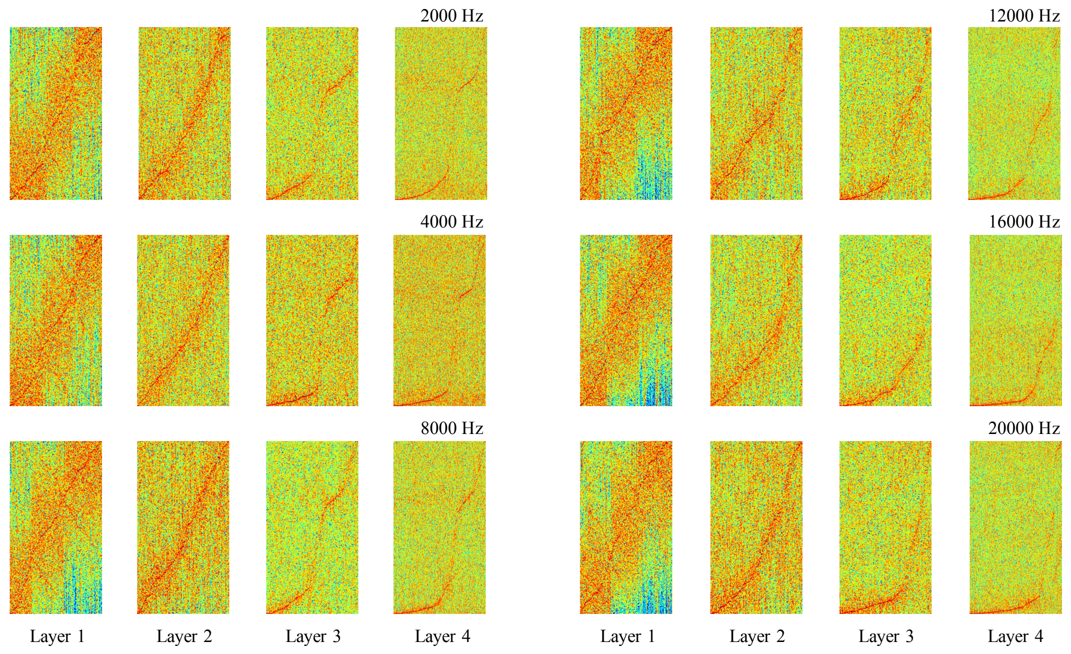


Figure 3.4: Spectrum visualization of learned filters for different sampling rates. The x-axis represents the index of the filter, and the y-axis represents the frequency ranged from 0 to half the sampling rate. 3-sized filters are used for the first four modules in all models for fairly visualizing learned filters.

Song-Level Similarity Using t-SNE

We extracted features from SampleCNN and aggregated them at different hierarchical levels of layer for each audio clip. We then embedded the song-level features into 2-D vectors using t-Distributed Stochastic Neighbor Embedding (t-SNE). Figure 3.5 visualizes the 2-D embedded features at different layer levels for selected tags to examine how multi-level feature aggregation technique enhances the performance. Songs with genre tag (*Techno*) are more closely clustered in the higher layer (-1 layer). On the other hand, songs with instrument tag (*Piano*) are more closely clustered in the lower layer (-3 layer). This may indicate that the optimal layer of feature representations can be different depending on the type of labels. Thus, combining different levels of features can improve the performance.

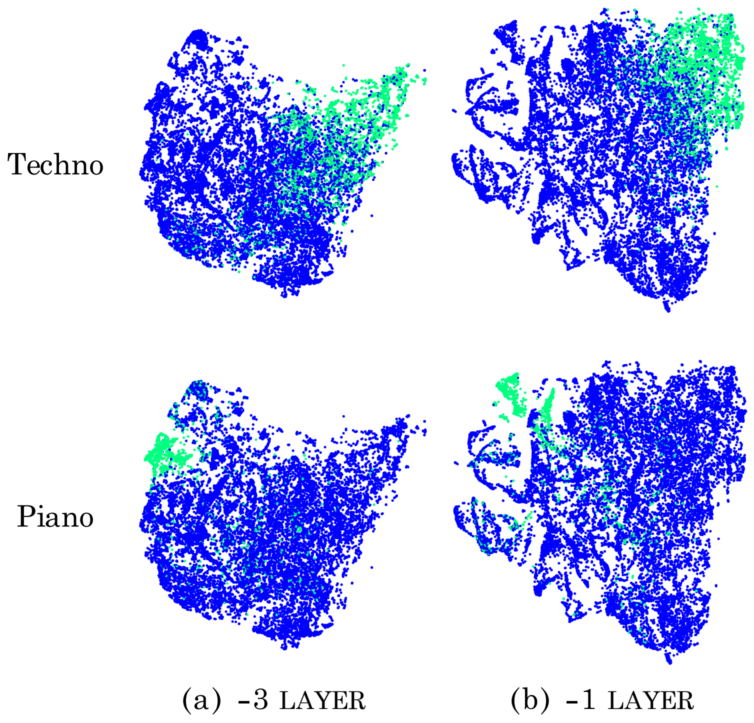


Figure 3.5: Feature visualization on songs with *Piano* tag and songs with *Techno* tag on MTAT using t-SNE. Features are extracted from (**a**) -3 LAYER and (**b**) -1 LAYER of the 3^9-SampleCNN model pre-trained with MSD.

3.1.8 Contribution Summary

In this article, we extend our previously proposed SampleCNN for music classification. Through the experiments, we found that downsampling music audio down to 8000 Hz does not significantly degrade performance but it saves training time. Second, transfer learning experiments with multi-level and multi-scale technique showed state-of-the-art results on most of the datasets we tested. Finally, we visualized the spectrum of the learned filters for each sampling rate and found that the SampleCNN model is actively focusing on (or zoom in on) important low-frequency bands. As future work, we will analyze why the sample-level architecture works well without input normalization and nonlinear function that compresses the amplitude, which are important when we use spectrogram as input. Also, we will investigate different filter visualization techniques to interpret the hierarchically-learned filters better.

3.2 Extensions to More Building Blocks and Audio Classification Tasks

Music, speech, and acoustic scene sound are often handled separately in the audio domain because of their different signal characteristics. However, as the image domain grows rapidly by versatile image classification models, it is necessary to study extensible classification models in the audio domain as well. In this study, we approach this problem using two types of sample-level deep convolutional neural networks that take raw waveforms as input and uses filters with small granularity. One is a basic model that consists of convolution and pooling layers. The other is an improved model that additionally has residual connections, squeeze-and-excitation modules and multi-level concatenation. We show that the sample-level models reach state-of-the-art performance levels for the three different categories of sound. Also, we visualize the filters along layers and compare the characteristics of learned filters.

3.2.1 Problem

Broadly speaking, audio classification tasks are divided into three sub-domains including music classification, speech recognition (particularly for the acoustic model), and acoustic scene classification. However, input audio features and models for each sub-domain task are usually different due to the different signal characteristics.

Recent advances in deep learning have encouraged a single audio classification model to be applied to many cross-domain tasks. For example, Lee et. al. proposed Convolutional Deep Belief Networks (CDBN) and applied it to phoneme classification, speaker identification, music genre and artist classification [51]. Adavanne et. al. used a Convolutional Recurrent Neural Network (CRNN) model for sound event detection [52], bird audio classification [53] and music emotion recognition [54]. However, the majority of the audio classification models use a 2D time-frequency representation as input, which involves different choices of time-frequency resolution, filter-bank size and magnitude compression.

This issue can be solved by a waveform-based model that directly takes raw input signals. Recently, Dieleman and Schrauwen used raw waveforms as input of CNN models for music auto-tagging task [12]. Sainath et. al. used Convolutional Long short-term memory Deep Neural Network (CLDNN) for speech recognition [18]. Dai et. al. used Deep Convolutional Neural Networks (DCNN) with residual connections for environmental sound recognition [23]. All of them used frame-level filters (typically several hundred samples long) in the first convolutional layer which were carefully designed to handle their target problems. In this type of architectures, the filters in the bottom layer should learn all possible phase variations of periodic waveforms which are likely to be prevalent in audio signals. However, the phase variations within a frame (i.e. time shift of periodic waveforms) are actually removed in the spectrogram.

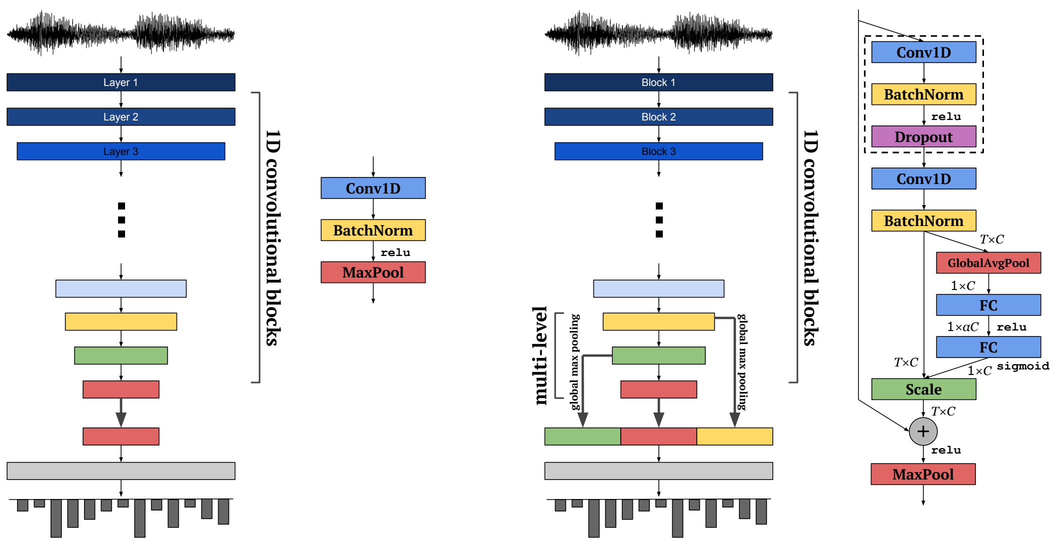
This problem is analogous to translation invariance in the image domain. Considering that filter size is typically small in the image domain, even 3×3 in the VGG model [26], we investigated the possibility of stacking very small-size filters from the bottom layer of DCNN for raw audio waveforms with max-pooling layers. The results on music auto-tagging [25] and sound event detection [55] showed that the VGG-style 1-D CNN models are highly effective. We term this model as **SampleCNN**.

We enhanced the SampleCNN model by adding residual connections, squeeze-and-excitation modules and multi-level feature concatenation for music auto-tagging [56]. The residual connection makes gradient propagation more fluent, allowing training deeper networks [57]. The Squeeze-and-Excitation (SE) module recalibrates filter-wise feature responses [58]. The multi-level feature concatenation takes different abstraction levels of classification labels into account [27]. We term this model as **ReSE-2-Multi**.

In this study, we show that the sample-level CNN models are effective for three datasets from different audio domains. Furthermore, we visualize hierarchically learned filters for each dataset in the waveform-based model to explain how they process sound differently.

3.2.2 Models

Figure 3.6 shows the structures of the two sample-level models. 2 or 3 sample-size 1D filters and poolings are used in all convolutional layers. In SampleCNN,



(a) SampleCNN (b) ReSE-2-Multi

Figure 3.6: SampleCNN and ReSE-2-Multi models.

convolutional layer, batch normalization layer and max-pooling layer are stacked as shown in Figure 3.6 (a). The detailed description can be found in [25].

In ReSE-2-Multi, we add a residual connection and an SE module onto the SampleCNN building block as shown in Figure 3.6 (b). The SE path recalibrates feature maps through two operations. One is squeeze operation that aggregates a global temporal information into filter-wise statistics using global average pooling. The operation reduces the temporal dimensionality (T) to one. The other is excitation operation that adaptively recalibrates each filter map using the filter-wise statistics from the squeeze operation and a simple gating mechanism. The gating consists of two fully-connected (FC) layers that compute nonlinear interactions among filters. Then, the original outputs from the basic block are rescaled by filter-wise multiplication with the sigmoid activation of the second FC layer of the SE path. We also added residual connections to train a deeper model. The digit in the model name, ReSE-2-Multi, indicates the number of convolution layers in one building block. Finally, we concatenate three hidden layers to take account of different levels of abstraction.

	Music	**Speech**	**Scene sound**
Dataset			
	MagnaTagATune (MTAT) [34]	Speech Commands Dataset [59, 60]	DCASE 2017 Task 4 [61]
		(TensorFlow Speech Recognition Challenge)	(subtask A)
Task	Music auto-tagging	Speech command recognition	Acoustic scene tagging
# of classes	50 tags	10 commands + "silence" + "unknown"	17 sound events
		(31 classes in training / 12 classes in testing)	
Labels	Multi-label	Multi-class	Multi-label
Sampling rate	22,050Hz	16,000Hz	44,100Hz
Dataset split	15,244 / 1529 / 4332	57,929 / 6798 / 30% of 158,538	45,313 / 5859 / 488
(train/valid/test)		(public leaderboard test setting)	(development set)
Duration	29 seconds	1 second	10 seconds
Description	Collected using the	Single-word speaking commands,	Subset of AudioSet [62],
	TagATune game and music	rather than conversational sentences	YouTube clips focusing on
	from Magnatune.		vehicle and warning sounds.
Model			
	(resampled to 16,000Hz)		(resampled to 16,000Hz)
Input size	39,366 samples, 2.46 sec	16,000 samples, 1 sec	19,683 samples, 1.23 sec
# of segments	12 segments per clip	1 segments per clip	9 segments per clip
# of blocks	9 blocks	8 blocks	8 blocks
Results			
	AUC	Accuracy	F-score (instance-based)
SampleCNN	0.9033	84%	38.9%
ReSE-2+Multi	0.9091	86%	45.1%
State-of-the-art	0.9113 [56]	88% (as of Nov 29, 2017) [60]	57.7% [63]

Table 3.7: Description of the three datasets, models and results.

3.2.3 Datasets and Results

We validate the effectiveness of the proposed models on music auto-tagging, speech command recognition and acoustic scene tagging. The details about the datasets for the tasks are summarized in Table 3.7. Note that we resampled all audio samples to 16,000Hz in order to verify how extensible the models are for the three sub-domain of audio tasks in the same condition. However, we configured the input size of the models for each dataset to commonly used size in each domain. Then, we set the number of building blocks according to the input size of the models. We averaged the prediction score for all segments of one clip in testing phase if the input size of the model is shorter than the duration of the clip.

Table 3.7 compares the results from the two sample-level CNN models. The performances are reported using commonly used evaluation metrics for each task. We also compare them to state-of-the-art performance on each dataset. In general, the ReSE-2-Multi model shows close performance to the state-of-the-art results except for the DCASE task. However, in [63], they used data balancing, ensemble networks and auto thresholding techniques. Without those techniques, they report that their CRNN model achieved 42.0% F-score value which is lower than our result with ReSE-2-Multi. Also, for the music auto-tagging task on MTAT, the state-of-the-art result was achieved by the ReSE-2-Multi model. In this case, the performance degradation is seen to be caused by downsampling to 16,000Hz.

3.2.4 Filter Visualizations

Visualizing the filters at each layer allows better understanding of representation learning in the hierarchical networks. Since both models yielded similar patterns of learned filters at each layer, we visualize them only for the sampleCNN model. Figure 3.7 shows the filters obtained by an activation maximization method [48]. To show the patterns more clearly, we visualized them as spectrum in the frequency domain and sorted them by the frequency with the peak magnitude [25]. In this case, we set the size of the initial random noise to 729 ($= 3^6$) samples, so that the estimated filters have typical frame-sized shape which will make the spectrum clearer. Also, for the first 6 layers we only used 3-sized filters and sub-sampling layers. Thus, the temporal dimension of the 6th layer output becomes one in this configuration. For other layers, we averaged remaining temporal dimension so as to make a single activation loss value. Finally, we conducted log-based magnitude compression on the spectrum.

From the figure, we can first find that they are sensitive to log-scaled frequency along the layers, such as mel-frequency spectrogram that is widely used in audio classification tasks. Second, when comparing acoustic scene sound with the other domains, the learned filters tend to have more low-frequency concentration and less complex patterns. This is probably because the DCASE task 4 dataset is made up of simple traffic and warning sounds. Finally, between music and speech, we can observe that more filters explain low-frequency content in music than speech.

3.2.5 Contribution Summary

In this paper, we evaluated the two sample-level CNN models on three datasets. The results show the possibility that they can be applied to different audio domains as a true end-to-end model. As future work, we will investigate more filter visualization techniques to have better understanding of the models.

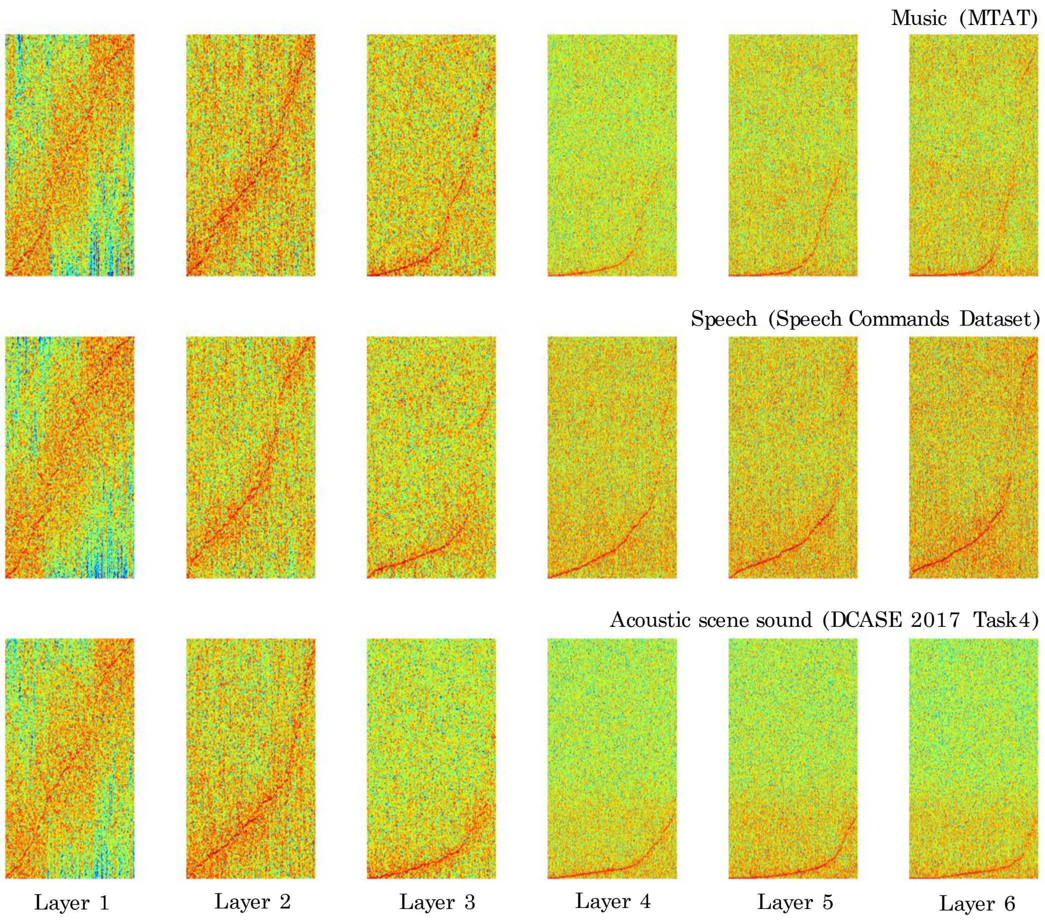


Figure 3.7: The spectrum of learned filter estimates for the three datasets in SampleCNN. They are sorted by the frequency with the peak magnitude. The x-axis represents the index of the filters and the y-axis represents the frequency (ranging from 0 to 8000Hz for all figures). The visualizations were obtained using a gradient ascent method that finds the input waveform that maximizes the activation of a filter at each layer.

Chapter 4. Representation Learning of Music Using Objective Metadata

In this chapter, we propose to use objective music metadata to train similarity-based deep learning model instead of tag labels. The objective metadata has advantages in terms of scalability and cost. Then, more music metadata are used to train the models such as album and track information. By exploring the use of different similarity notions to train the model, we can see the effects and characteristics of different similarity notions.

4.1 Representation Learning of Music Using Artist Labels

In music domain, feature learning has been conducted mainly in two ways: unsupervised learning based on sparse representations or supervised learning by semantic labels such as music genre. However, finding discriminative features in an unsupervised way is challenging and supervised feature learning using semantic labels may involve noisy or expensive annotation. In this paper, we present a supervised feature learning approach using artist labels annotated in every single track as objective meta data. We propose two deep convolutional neural networks (DCNN) to learn the deep artist features. One is a plain DCNN trained with the whole artist labels simultaneously, and the other is a Siamese DCNN trained with a subset of the artist labels based on the artist identity. We apply the trained models to music classification and retrieval tasks in transfer learning settings. The results show that our approach is comparable to previous state-of-the-art methods, indicating that the proposed approach captures general music audio features as much as the models learned with semantic labels. Also, we discuss the advantages and disadvantages of the two models.

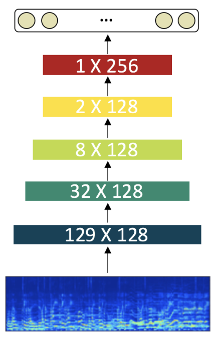
4.1.1 Problem

Representation learning or feature learning has been actively explored in recent years as an alternative to feature engineering [64]. The data-driven approach, particularly using deep neural networks, has been applied to the area of music information retrieval (MIR) as well [65]. In this paper, we propose a novel audio feature learning method using deep convolutional neural networks and artist labels.

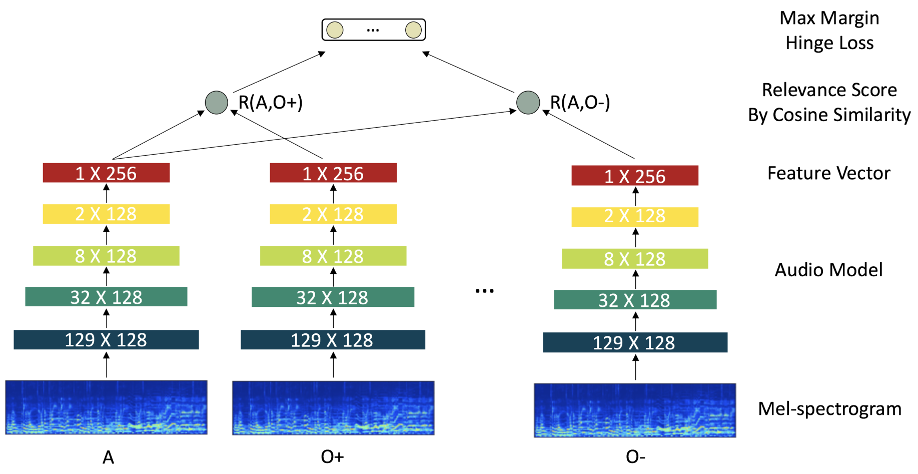
Early feature learning approaches are mainly based on unsupervised learning algorithms. Lee et al. used convolutional deep belief network to learn structured acoustic patterns from spectrogram [51]. They showed that the learned features achieve higher performance than Mel-Frequency Cepstral Coefficients (MFCC) in genre and artist classification. Since then, researchers have applied various unsupervised learning algorithms such as sparse coding [5, 66–68], K-means [5, 38, 69] and restricted Boltzmann machine [5,70]. Most of them focused on learning a meaningful dictionary on spectrogram by exploiting sparsity. While these unsupervised learning approaches are promising in that it can exploit abundant unlabeled audio data, most of them are limited to single or dual layers, which are not sufficient to represent complicated feature hierarchy in music.

On the other hand, supervised feature learning has been progressively more explored. An early approach was mapping a single frame of spectrogram to genre or mood labels via pre-trained deep neural networks and using the hidden-unit activations as audio features [71, 72]. More recently, this approach was handled in the context of transfer learning using deep convolutional neural networks (DCNN) [73, 74]. Leveraging large-scaled datasets and recent advances in deep learning, they showed that the hierarchically learned features can be effective for diverse music classification tasks. However, the semantic labels that they use such as genre, mood or other timbre descriptions tend to be noisy as they are sometimes ambiguous to annotate or tagged from the crowd. Also, high-quality annotation by music experts is known to be highly time-consuming and expensive.

Meanwhile, artist labels are the meta data annotated to songs naturally from the album release. They are objective information with no disagreement. Furthermore, considering every artist has his/her own style of music, artist labels may be regarded



(a) The Basic Model



(b) The Siamese Model

Figure 4.1: The proposed architectures for the model using artist labels.

as terms that describe diverse styles of music. Thus, if we have a model that can discriminate different artists from music, the model can be assumed to explain various characteristics of the music.

In this paper, we verify the hypothesis using two DCNN models that are trained to identify the artist from an audio track. One is the basic DCNN model where the softmax output units corresponds to each of artist. The other is the Siamese DCNN trained with a subset of the artist labels to mitigate the excessive size of the output layer in the plain DCNN when a large-scale dataset is used. After training the two models, we regard them as a feature extractor and apply artist features to three different genre datasets in two experiment settings. First, we directly find similar songs using the artist features and K-nearest neighbors. Second, we conduct transfer learning to further adapter the features to each of the datasets. The results show that proposed approach captures useful features for unseen audio datasets and the propose models are comparable to those trained with semantic labels in performance. In addition, we discuss the advantages and disadvantages of the two proposed DCNN models.

4.1.2 Learning Models

Figure 4.1 shows the two proposed DCNN models to learn audio features using artist labels. The basic model is trained as a standard classification problem. The Siamese model is trained using pair-wise similarity between an anchor artist and other artists. In this section, we describe them in detail.

Basic Model

This is a widely used 1D-CNN model for music classification [8, 12, 31, 74]. The model uses mel-spectrogram with 128 bins in the input layer. We configured the DCNN such that one-dimensional convolution layers slide over only a single temporal dimension. The model is composed of 5 convolution and max pooling layers as illustrated in Figure 4.1(a). Batch normalization [37] and rectified linear unit (ReLU) activation layer are used after every convolution layer. Finally, we used categorical cross entropy loss in the prediction layer. We train the model to classify artists instead of semantic labels used in many music classification tasks. For example, if the number of artists used is 1,000, this becomes a classification problem that identifies one of the 1,000 artists. After training, the extracted 256-dimensional feature vector in the last hidden layer is used as the final audio feature learned using artist labels. Since this is the representation from which the identity is predicted by the linear softmax classifier, we can regard it as the highest-level artist feature.

Siamese Model

While the basic model is simple to train, it has two main limitations. One is that the output layer can be excessively large if the dataset has numerous artists. For example, if a dataset has 10,000 artists and the last hidden layer size is 100, the number of parameters to learn in the last weight matrix will reach 1M. Second, whenever new artists are added to the dataset, the model must be trained again entirely. We solve the limitations using the Siamese DCNN model.

A Siamese neural network consists of twin networks that share weights and configuration. It then provides unique inputs to the network and optimizes similarity scores [75–77]. This architecture can be extended to use both positive and negative examples at one optimization step. It is set up to take three examples: anchor item (query song), positive item (relevant song to the query) and negative item (different song to the query). This model is often called *triplet networks* and has been successfully applied to music metric learning when the relative similarity scores of song triplets are available [78]. This model can be further extended to use several negative samples instead of just one negative in the triplet network. This technique is called *negative sampling* and has been popularly used in word embedding [15] and latent semantic model [79]. By using this technique, they could effectively approximate the full softmax function when the output class is extremely large (i.e. 10,000 classes).

We approximate the full softmax output in the basic model with the Siamese neural networks using negative sampling technique. Regarding the artist labels, we set up the negative sampling by treating identical artist's song to the anchor song as positive sample and other artists' songs as negative samples. This method is illustrated in Figure 4.1(b). Following [79], the relevance score between the anchor song feature and other song feature is measured as:

$$R(A, O) = \cos(y_A, y_O) = \frac{y_A^T y_O}{|y_A||y_O|} \tag{4.1}$$

where y_A and y_O are the feature vectors of the anchor song and other song, respectively.

Meanwhile, the choice of loss function is important in this setting. We tested two loss functions. One is the softmax function with categorical cross-entropy loss to maximize the positive relationships. The other is the max-margin hinge loss to set only margins between positive and negative examples [80]. In our preliminary experiments, the Siamese model with negative sampling was successfully trained only with the max-margin loss function between the two objectives, which is defined as follows:

$$loss(A, O) = \sum_{O^-} \max[0, \Delta - R(A, O^+) + R(A, O^-)] \tag{4.2}$$

where Δ is the margin, O^+ and O^- denotes positive example and negative examples, respectively. We also grid-searched the number of negative samples and the margin, and finally set the number of negative samples to 4 and the margin value Δ to 0.4. The shared audio model used in this approach is exactly the same configuration as the basic model.

Compared Model

In order to verify the usefulness of the artist labels and the presented models, we constructed another model that has the same architecture as the basic model but using semantic tags. In this model, the output layer size corresponds to the number of the tag labels. Hereafter, we categorize all of them into *artist-label model* and *tag-label model*, and compare the performance.

4.1.3 Experiments

In this section, we describe source datasets to train the two artist-label models and one tag-label model. We also introduce target datasets for evaluating the three models. Finally, the training details are explained.

Source Tasks

All models are trained with the Million Song Dataset (MSD) [36] along with 30-second 7digital[1] preview clips. Artist labels are naturally annotated onto every song, thus we simply used them. For the tag label, we used the Last.fm dataset augmented on MSD. This dataset contains tag annotation that matches the ID of the MSD.

[1]https://www.7digital.com/

Artist-label Model

The number of songs that belongs to each artist may be extremely skewed and this can make fair comparison among the three models difficult. Thus, we selected 20 songs for each artist evenly and filtered out the artists who have less than this. Also, we configured several sets of the artist lists to see the effect of the number of artists on the model performances (500, 1,000, 2,000, 5,000 and 10,000 artists). We then divided them into 15, 3 and 2 songs for training, validation and testing, respectively for the sets contain less than 10,000 artists. For the 10,000 artist sets, we partitioned them in 17, 1 and 2 songs because once the artists reach 10,000, the validation set already become 10,000 songs even when we only use 1 song from each artist which is already sufficient for validating the model performance. We also should note that the testing set is actually not used in the whole experiments in this paper because we used the source dataset only for training the models to use them as feature extractors. The reason we filtered and split the data in this way is for future work[2].

Tag-label Model

We used 5,000 artists set as a baseline experiment setting. This contains total 90,000 songs in the training and validation set with a split of 75,000 and 15,000. We thus constructed the same size set for tagging dataset to compare the artist-label models and the tag-label model. The tags and songs are first filtered in the same way as the previous works [7, 74]. Among the list with the filtered top 50 used tags, we randomly selected 90,000 songs and split them into the same size as the 5,000 artist set.

Target Tasks

We used 3 different datasets for genre classification.

- GTZAN (fault-filtered version) [32,33]: 930 songs, 10 genres. We used a "fault-filtered" version of GTZAN [33] where the dataset was divided to prevent artist repetition in training/validation/test sets.

[2]All the data splits of the source tasks are available at the link for reproducible research https://github.com/jiyoungpark527/msd-artist-split.

- FMA small [1]: 8,000 songs, 8 balanced genres.
- NAVER Music[3] dataset with only Korean artists: 8,000 songs, 8 balanced genres. We filtered songs with only have one genre to clarify the genre characteristic.

Training Details

For the preprocessing, we computed the spectrogram using 1024 samples for FFT with a Hanning window, 512 samples for hop size and 22050 Hz as sampling rate. We then converted it to mel-spectrogram with 128 bins along with a log magnitude compression.

We chose 3 seconds as a context window of the DCNN input after a set of experiments to find an optimal length that works well in music classification task. Out of the 30-second long audio, we randomly extracted the context size audio and put them into the networks as a single example. The input normalization was performed by dividing standard deviation after subtracting mean value across the training data.

We optimized the loss using stochastic gradient descent with 0.9 Nesterov momentum with $1e^{-6}$ learning rate decay. Dropout 0.5 is applied to the output of the last activation layer for all the models. We reduce the learning rate when a valid loss has stopped decreasing with the initial learning rate 0.015 for the basic models (both artist-label and tag-label) and 0.1 for the Siamese model. Zero-padding is applied to each convolution layer to maintain its size.

Our system was implemented in Python 2.7, Keras 2.1.1 and Tensorflow-gpu 1.4.0 for the back-end of Keras. We used NVIDIA Tesla M40 GPU machines for training our models. Code and models are available at the link for reproducible research[4].

4.1.4 Feature Evaluation

We apply the learned audio features to genre classification as a target task in two different approaches: feature similarity-based retrieval and transfer learning. In this

[3]http://music.naver.com

[4]https://github.com/jongpillee/ismir2018-artist.

section, we describe feature extraction and feature evaluation methods.

Feature Extraction Using the DCNN Models

In this work, the models are evaluated in three song-level genre classification tasks. Thus, we divided 30-second audio clip into 10 segments to match up with the model input size and the 256-dimension features from the last hidden layer are averaged into a single song-level feature vector and used for the following tasks. For the tasks that require song-to-song distances, cosine similarity is used to match up with the Siamese model's relevance score.

Feature Similarity-based Song Retrieval

We first evaluated the models using mean average precision (MAP) considering genre labels as relevant items. After obtaining a ranked list for each song based on cosine similarity, we measured the MAP as following:

$$AP = \frac{\sum_{k \in rel} precision_k}{number\ of\ relevant\ items} \tag{4.3}$$

$$MAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \tag{4.4}$$

where Q is the number of queries. $precision_k$ measures the fraction of correct items among first *k* retrieved list.

The purpose of this experiment is to directly verify how similar feature vectors with the same genre are in the learned feature space.

Transfer Learning

We classified audio examples using the k-nearest neighbors (k-NN) classifier and linear softmax classifier. The evaluation metric for this experiment is classification accuracy. We first classified audio examples using k-NN to classify the input audio into the largest number of genres among k nearest to features from the training set. The number of k is set to 20 in this experiment. This method can be regarded as a similarity-based classification. We also classified audio using a linear softmax clas-

MAP	Artist-label Basic Model	Artist-label Siamese Model	Tag-label Model
GTZAN (fault-filtered)	0.4968	0.5510	0.5508
FMA small	0.2441	0.3203	0.3019
NAVER Korean	0.3152	0.3577	0.3576

Table 4.1: MAP results on feature similarity-based retrieval.

KNN	Artist-label Basic Model	Artist-label Siamese Model	Tag-label Model
GTZAN (fault-filtered)	0.6655	0.6966	0.6759
FMA small	0.5269	0.5732	0.5332
NAVER Korean	0.6671	0.6393	0.6898

Table 4.2: KNN similarity-based classification accuracy.

Linear Softmax	Artist-label Basic Model	Artist-label Siamese Model	Tag-label Model
GTZAN (fault-filtered)	0.6721	0.6993	0.7072
FMA small	0.5791	0.5483	0.5641
NAVER Korean	0.6696	0.6623	0.6755

Table 4.3: Classification accuracy of a linear softmax.

sifier. The purpose of this experiment is to verify how much the audio features of unseen datasets are linearly separable in the learned feature space.

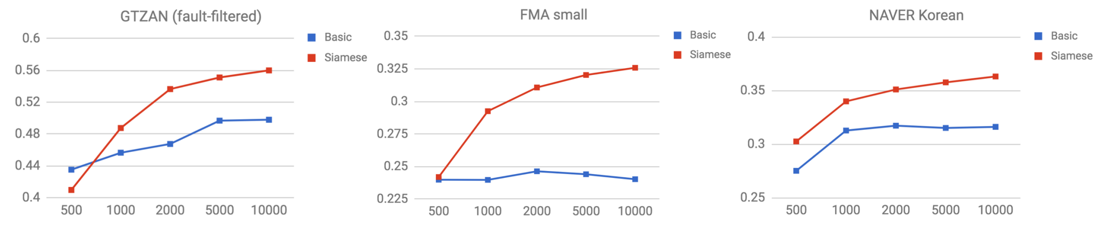


Figure 4.2: MAP results with regard to different number of artists in the feature models.

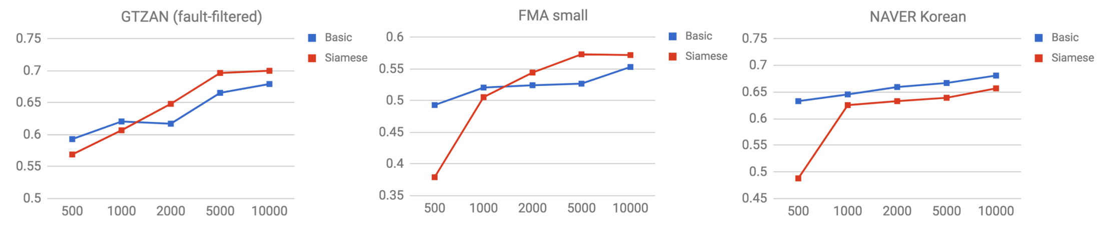


Figure 4.3: Genre classification accuracy using k-NN with regard to different number of artists in the feature models.

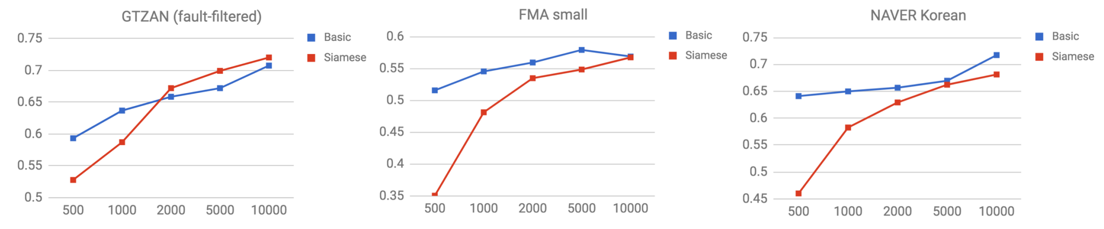


Figure 4.4: Genre classification accuracy using linear softmax with regard to different number of artists in the feature models.

4.1.5 Results and Discussion

Tag-label Model vs. Artist-label Model

We first compare the artist-label models to the tag-label model when they are trained with the same dataset size (90,000 songs). The results are shown in Table 4.1, 4.2 and 4.3. In feature similarity-based retrieval using MAP (Table 4.1), the artist-based Siamese model outperforms the rest on all target datasets. In the genre classification tasks (Table 4.2 and 4.3), Tag-label model works slightly better than the rest on some datasets and the trend becomes stronger in the classification using the linear softmax. Considering that the source task in the tag-based model (trained with the Last.fm tags) contains genre labels mainly, this result may attribute to the similarity of labels in both source and target tasks. Therefore, we can draw two conclusions from this experiment. First, the artist-label model is more effective in similarity-based tasks (4.1 and 4.2) when it is trained with the proposed Siamese networks, and thus it may be more useful for music retrieval. Second, the semantic-based model is more effective in genre or other semantic label tasks and thus it may be more useful for human-friendly music content organization.

Basic Model vs. Siamese Model

Now we focus on the comparison of the two artist-label models. From Table 4.1, 4.2 and 4.3, we can see that the Siamese model generally outperforms the basic model. However, the difference become attenuated in classification tasks and the Siamese model is even worse on some datasets. Among them, it is notable that the Siamese model is significantly worse than the basic model on the NAVER Music dataset in the genre classification using k-NN even though they are based on feature similarity. We dissected the result to see whether it is related to the cultural difference between the training data (MSD, mostly Western) and the target data (the NAVER set, only Korean). Figure 4.5 shows the detailed classification accuracy for each genre of the NAVER dataset. In three genres, ‘Trot’,‘K-pop Ballad’ and ‘Kids’ that do not exist in the training dataset, we can see that the basic model outperforms the Siamese model whereas the results are opposite in the other genres. This indicates that the basic model is more robust to unseen genres of music. On the other hand, the Siamese model slightly over-fits to the training set, although it effectively captures the artist features.

Effect of the Number of Artists

We further analyze the artist-label models by investigating how the number of artists in training the DCNN affects the performance. Figure 4.2, 4.3 and 4.4 are the results that show similarity-based retrieval (MAP) and genre classification (accuracy) using k-NN and linear softmax, respectively, according to the increasing number of training artists. They show that the performance is generally proportional to the number of artists but the trends are quite different between the two models. In the

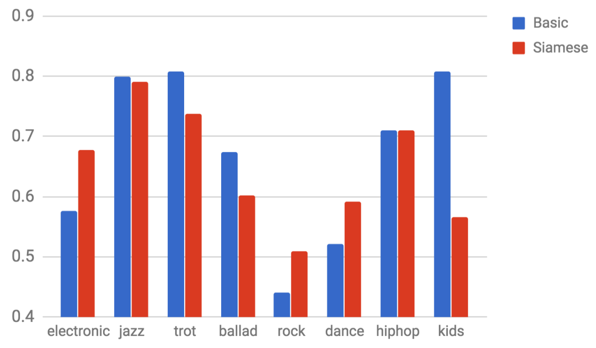


Figure 4.5: The classification results of each genre for the NAVER dataset with only Korean music.

Models	GTZAN (fault-filtered)	FMA small
2-D CNN [33]	0.6320	-
Temporal features [43]	0.6590	-
Multi-level Multi-scale [74]	0.7200	-
SVM [1]	-	0.5482†
Artist-label Basic model	0.7076	0.5687
Artist-label Siamese model	0.7203	0.5673

Table 4.4: Comparison with previous state-of-the-art models: classification accuracy results. Linear softmax classifier is used and features are extracted from the artist-label models trained with 10,000 artists. † This result was obtained using the provided code and dataset in [1].

similarity-based retrieval, the MAP of the Siamese model is significantly higher than that of the basic model when the number of artists is greater than 1,000. Also, as the number of artists increases, the MAP of the Siamese model consistently goes up with a slight lower speed whereas that of the basic model saturates at 2,000 or 5,000 artists. On the other hand, the performance gap changes in the two classification tasks. On the GTZAN dataset, while the basic model is better for 500 and 1,000 artists, the Siamese model reverses it for 2,000 and more artists. On the NAVER dataset, the basic model is consistently better. On the FMA small, the results are mixed in two classifiers. Again, the results may be explained by our interpretation of the models in Section 4.1.5. In summary, the Siamese model seems to work better in similarity-based tasks and the basic model is more robust to different genres of music. In addition, the Siamese model is more capable of being trained with a large number of artists.

Comparison with State-of-the-arts

The effectiveness of artist labels is also supported by comparison with previous state-of-the-art models in Table 4.4. For this result, we report two artist-label models trained with 10,000 artists using linear softmax classifier. In this table, we can see that the proposed models are comparable to the previous state-of-the-art methods.

4.1.6 Visualization

We visualize the extracted feature to provide better insight on the discriminative power of learned features using artist labels. We used the DCNN trained to classify 5,000 artists as a feature extractor. After collecting the feature vectors, we embedded them into 2-dimensional vectors using t-distributed stochastic neighbor embedding (t-SNE).

For artist visualization, we collect a subset of MSD (apart from the training data for the DCNN) from well-known artists. Figure 4.6 shows that artists' songs are appropriately distributed based on genre, vocal style and gender. For example, artists with similar genre of music are closely located and female pop singers are close to each other except Maria Callas who is a classical opera singer. Interestingly, some songs by Michael Jackson are close to female vocals because of his distinctive high-pitched tone.

Figure 4.7 shows the visualization of features extracted from the GTZAN dataset. Even though the DCNN was trained to discriminate artist labels, they are well clustered by genre. Also, we can observe that some genres such as disco, rock and hip-hop are divided into two or more groups that might belong to different sub-genres.

4.1.7 Contribution Summary and Future work

In this work, we presented the models to learn audio feature representation using artist labels instead of semantic labels. We compared two artist-label models and one

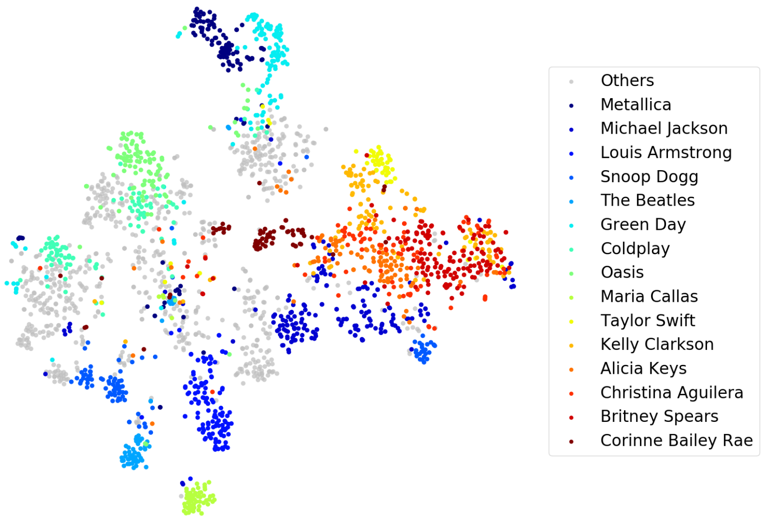


Figure 4.6: Feature visualization by artist. Total 22 artists are used and, among them, 15 artists are represented in color.

tag-label model. The first is a basic DCNN consisting of a softmax output layer to predict which artist they belong to out of all artists used. The second is a Siamese-style architecture that maximizes the relative similarity score between a small subset of the artist labels based on the artist identity. The last is a model optimized using tag labels with the same architecture as the first model. After the models are trained, we used them as feature extractors and validated the models on song retrieval and genre classification tasks on three different datasets. Three interesting results were found during the experiments. First, the artist-label models, particularly the Siamese model, is comparable to or outperform the tag-label model. This indicates that the cost-free artist-label is as effective as the expensive and possibly noisy tag-label. Second, the Siamese model showed the best performances on song retrieval task in all datasets tested. This can indicate that the pair-wise relevance score loss in the Siamese model helps the feature similarity-based search. Third, the use of a large number of artists increases the model performance. This result is also useful because the artists can be easily increased to a very large number.

As future work, we will investigate the artist-label Siamese model more thoroughly. First, we plan to investigate advanced audio model architecture and diverse loss and pair-wise relevance score functions. Second, the model can easily be re-trained using new added artists because the model does not have fixed output layer.

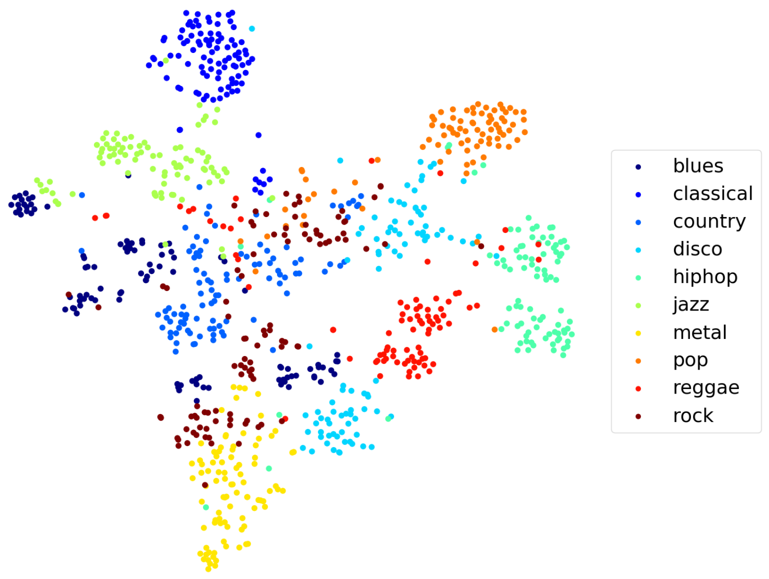


Figure 4.7: Feature visualization by genre. Total 10 genres from the GTZAN dataset are used.

This property will be evaluated using cross-cultural data or using extremely small data (i.e. one-shot learning [76]).

4.2 Representation Learning of Music using Artist, Album, and Track Information

Supervised music representation learning has been performed mainly using semantic labels such as music genres. However, annotating music with semantic labels requires time and cost. In this work, we investigate the use of factual metadata such as artist, album, and track information, which are naturally annotated to songs, for supervised music representation learning. The results show that each of the metadata has individual concept characteristics, and using them jointly improves overall performance.

4.2.1 Problem

Representation learning of music has been recently performed by supervised deep learning using semantic labels such as genres, moods and instruments [27, 73]. However, annotating music with such semantic labels requires significant time and cost and the labels are often ambiguous, resulting in disagreement among annotators [81]. Meanwhile, metadata such as artist labels require no cost and they are factual information with no ambiguity. We recently investigated the possibility of using artist information for representation learning of music and evaluated it in transfer learning settings [82]. The results showed that the learned representation is comparable to those using the semantic labels. In this work, we extend the use of music metadata to album and track information, which are more specific levels than the artist information. We use a similarity-based learning model following the previous work and also report the effects of the number of negative samples and training samples.

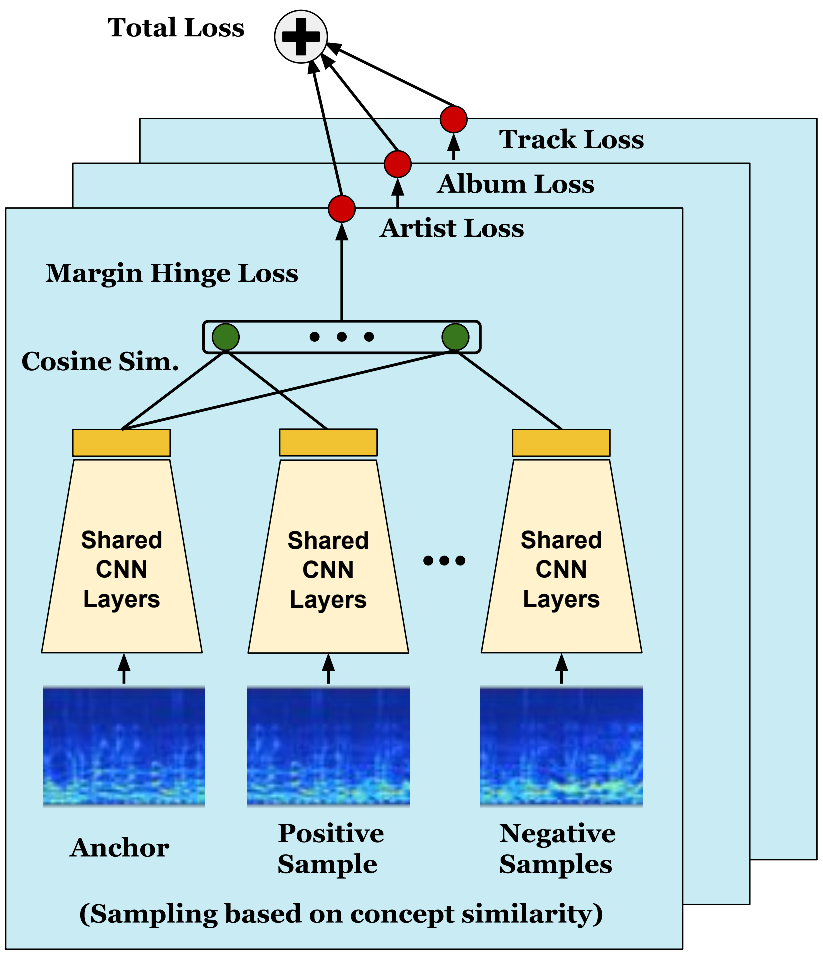


Figure 4.8: Joint learning model using artist, album, and track information.

Learned concept	Artist	Album	Track	Artist +Album +Track
Artist	0.680	0.634	0.539	0.686
Album	0.732	0.822	0.653	0.763
Track	0.922	0.958	0.971	0.945

Table 4.5: Hold-out positive and negative sample prediction.

4.2.2 Models

Figure 4.8 illustrates the overview of representation learning model using artist, album, and track information. Following the previous work, we use a Siamese-style Convolutional Neural Network (CNN) with multiple negative samples[5]. We build one large model that jointly learns artist, album, and track information and three single models that learns each of artist, album, and track information separately for comparison. The single model basically takes anchor sample, positive sample, and negative samples based on the similarity notion. For example, in the artist similarity concept, positive and negative samples are selected based on whether the sample is from the same artist as the anchor sample. We should note that the model takes a segment of audio (e.g. 3 second long), not the whole chunk of the song audio. Thus, in the track similarity concept, positive and negative samples are chosen based on whether the sample segment is from the same track as the anchor segment. Finally, we construct a joint learning model by simply adding three loss functions from the three similarity concepts, and share model parameters for all of them.

Learned concept	genres (baseline)	Artist	Album	Track	Artist +Album +Track
GTZAN	0.547	0.724	0.652	0.564	0.745
FMA small	0.533	0.598	0.560	0.463	0.593
NAVER Korean	0.720	0.662	0.641	0.549	0.663

Table 4.6: Transfer learning experiment. Baseline results are generated by performing genre classification directly without transfer learning.

Number of Negative Samples	1	2	4	8	16
GTZAN	0.665	0.663	0.681	0.702	0.711
FMA small	0.544	0.535	0.568	0.573	0.578
NAVER Korean	0.643	0.634	0.658	0.676	0.673

Table 4.7: The effect of the number of negative samples. The model is trained with 1000 artists, 2000 albums, and its related track concepts.

4.2.3 Experiments and Evaluations

The four models are trained with Million Song Dataset (MSD) and its artist and album metadata [36]. We first build two splits based on each artist and album information. The artist split is the same as the previous work, which has 20 songs for each artist. For the album split, we selected 10 songs for each album and used twice as many albums to match the number of training samples of artist. Then, 10 songs of one album are divided into 8 songs, 1 song, and 1 song for training, validation and testing. The artist split is twice these numbers. For the validation sampling of artist or

[5] In this work, we used twice the number of filters for all layers.

Number of Training Artists	500	1000	2000	5000	10000
GTZAN	0.638	0.681	0.706	0.745	0.755
FMA small	0.517	0.568	0.588	0.593	0.603
NAVER Korean	0.636	0.658	0.668	0.663	0.686

Table 4.8: The effect of the number of training samples. The model is trained with 4 negative samples with artist, album, and track concepts. The number of albums used is twice the number of artists.

album concept, the positive sample is selected from the training set and the negative samples are chosen from the validation set based on the validation anchor's concept. For the track concept, it basically follows the artist split, and the positive sample for the validation sampling is chosen from the other part of the anchor song.

The evaluation is conducted in two ways: 1) hold-out positive and negative sample prediction and 2) transfer learning experiment. The hold-out positive and negative sample prediction was designed to see how well the models distinguish each concept. The evaluation is conducted on the test set of the above splits. For the artist and album concept, the positive sample is selected from the validation set and the negative samples are from test set based on the anchor's concept. In this evaluation, the random guess is 20% when the model uses 4 negative samples. The transfer learning experiment is performed on three external genre classification datasets including GTZAN (a fault-filtered version) [32, 33], FMA small [1], and NAVER Korean [82]. In this experiment, the learned representation is extracted and injected into a linear softmax classifier. This experiment was designed to see the generalization ability of the learned representations. For both evaluations, we used a model trained with 5000 artists (or/and 10000 albums) with 4 negative samples. After grid search, the margin values of loss function were set to 0.4, 0.25, and 0.1 for artist, album, and track concepts, respectively.

4.2.4 Results

The result of hold-out positive and negative sample prediction is shown in Table 4.5. We can see that each of the models performs best when the concept matches between the training set and test set. Also, the jointly learned model achieves good performance for all concepts.

The transfer learning experiment result is shown in Table 4.6. The artist model shows the best performance among the three single concept models, followed by the album model. This is probably because the genre classification task is more similar to the artist concept discrimination than album or track. The jointly learned model slightly outperforms the artist model. Finally, we included the baseline results obtained by performing genre classification directly without transfer learning. The results show that transfer learning using large music corpora with the factual metadata is highly effective in the GTZAN and FMA datasets, but not in NAVER dataset. This was due to the cross-cultural differences between the source and target datasets when looking closely at class-wise performances.

The effects of the number of negative samples and the number of training samples are shown in Table 4.7 and Table 4.8, respectively. We can see that increasing the number of negative samples and the number of training songs improves the model performance as expected.

Chapter 5. Disentangled Multidimensional Music Similarity

In this chapter, we opens up two new music search methdos which are query-by-attribute and query-by-prototype. To do that, we adopt Conditional Similarity Networks (CSN) to music similarity learning, and the concept of Prototypical Networks to music search applications. In addition, we propose a unified framework for music representation learning.

5.1 Metric Learning-based Approach

Music similarity search is useful for a variety of creative tasks such as replacing one music recording with another recording with a similar “feel”, a common task in video editing. For this task, it is typically necessary to define a similarity metric to compare one recording to another. Music similarity, however, is hard to define and depends on multiple simultaneous notions of similarity (i.e. genre, mood, instrument, tempo). While prior work ignore this issue, we embrace this idea and introduce the concept of multidimensional similarity and unify both global and specialized similarity metrics into a single, semantically disentangled multidimensional similarity metric. To do so, we adapt a variant of deep metric learning called conditional similarity networks to the audio domain and extend it using track-based information to control the specificity of our model. We evaluate our method and show that our single, multidimensional model outperforms both specialized similarity spaces and alternative baselines. We also run a user-study and show that our approach is favored by human annotators as well.

5.1.1 Problem

Traditional music *search* methods such as those available on streaming services and online music repositories use text-based metadata (e.g. song, artist, album, and/or semantic tags) for music retrieval. However, there are scenarios where music metadata is either unavailable or insufficient: a concrete example is what we shall refer to as

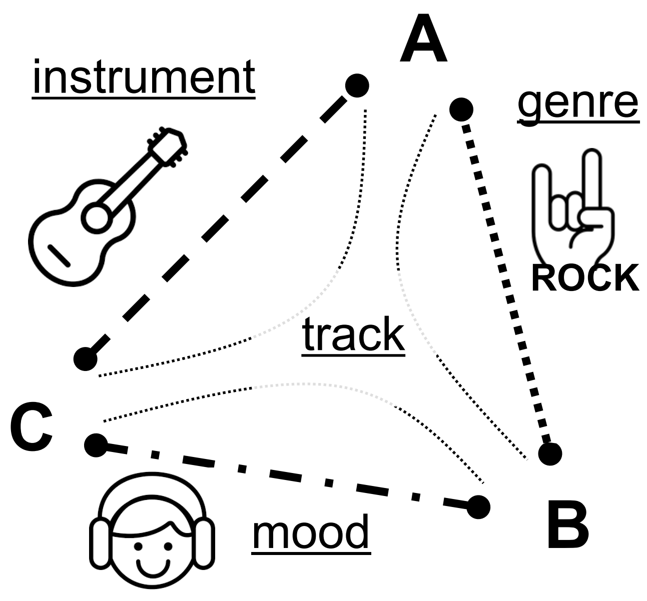


Figure 5.1: An illustration of multiple dimensions of music similarity. Letters (A, B, C) denote different music recordings, while lines denote different dimensions of similarity.

the "music replacement" problem, where a user wishes to replace one music recording with another recording that has a similar "feel", a common use case e.g. in video editing. Describing the desired musical traits may be extremely hard to do with text, but the user has an example of what they are searching for, and so query-by-example, and more specifically content-based music similarity and retrieval, is an attractive solution. While content-based music similarity has been studied extensively [2], it has found limited application in music recommendation platforms, which rely most heavily on interaction and metadata based collaborative filtering [3]. Such techniques are not applicable, however to the music replacement scenario, where there may be little-to-no interaction data, and a user's past music replacement selections can have little correlation with future replacement needs.

From a retrieval specificity perspective, music replacement is less specific than music identification (fingerprinting), but more specific than tag-based retrieval (e.g. genre) or than finding similar-sounding music for listening purposes [2, 4], since the pragmatic goal of music replacement is to find songs which sound as close as possible to a query without being identical.

Content-based music similarity typically involves extracting a feature representation from audio recordings and computing the similarity (or distance) between them using a metric or score function. Previous approaches include vector quantization [83], linear metric learning [9, 10, 84], and, more recently, deep metric learning [85–87] using human similarity labels [78], artist labels [82], track labels [88], or tags in the context of zero-shot learning [89]. A common limitation of these approaches is that similarity is modeled as uni-dimensional, i.e. songs are modelled as similar or dissimilar along a single global axis. In actuality, music is a multidimensional phenomenon, and consequently there are various *different* dimensions along which songs can be compared (e.g. timbre, rhythm, genre, mood, etc.), and songs can be simultaneously similar along some dimensions, while different along others, as illustrated in Figure 5.1. It is also hard to determine precisely which dimensions people take into account when rating songs for similarity, or how they weight the importance of these dimensions. For this reason, from an application standpoint it can be beneficial to allow the user to specify which musical dimensions they care about when searching-by-example and how to weight their importance.

In this paper, we propose a deep *disentangled* metric learning method for learning a multidimensional music similarity space (embedding). We adapt Conditional Similarity Networks [11], previously only applied to images, to the audio domain, and employ a combination of user-generated tags and algorithmic estimates (i.e. tempo) to train a disentangled embedding space composed of sub-spaces corresponding to similarity along different musical dimensions: genre, mood, instrumentation and tempo. Further, we propose a track-regularization technique to increase overall perceptual similarity across all dimensions as judged by humans. We evaluate our approach against several baselines, showing our proposed approach outperforms them both in terms of global similarity and similarity along specific dimensions. To validate our quantitative results, we run a user-study and show that our proposed approach is favored by human annotators as well.

5.1.2 Learning Model

Metric learning with triplet loss

We use deep metric learning with a triplet loss as the basis for our learning model [85,86]. On a high level, our model is presented with a triplet of samples, where one is considered the "anchor" and the other two consist of a "positive" and a "negative", and the model is trained to map the samples into an embedding space where the "positive" is closer to the "anchor" than the "negative", as illustrated in Figure 5.2 (A).

Formally, we define training triplets as a set $T = \{t^i\}_{i=1}^{N}$, where each triplet $t^i = \{x_a^i, x_p^i, x_n^i | s(x_a, x_p) > s(x_a, x_n)\}$, x_a is the anchor sample, x_p is the positive sample, x_n is the negative sample, and s is the musical dimension along which similarity is measured. Then, we define the triplet loss as:

$$L(t) = \max\{0, D(x_a, x_p) - D(x_a, x_n) + \Delta\}, \tag{5.1}$$

where $D(x_i, x_j) = ||f(x_i) - f(x_j)||_2$ is the euclidean distance between two audio embeddings, Δ is a margin value to prevent trivial solutions, and $f(\cdot)$ is a nonlinear embedding function or deep neural network that maps the audio input to the embedding space. For a given set T and embedding function $f(\cdot)$, we use stochastic gradient descent to update the network weights and minimize the loss.

Disentangling the embedding features

To jointly model multiple semantic dimensions of similarity within a single network, we adapt the work of Veit et al. [11], which proposed the use of Conditional Similarity Networks (CSN) [11] for attribute-based image retrieval. The method introduces masking functions $m_s \in \mathbb{R}^d$, which are applied to the embedding space of size d. Each mask corresponds to a certain similarity dimension s (denoted "condition" in [11]), e.g. mood or tempo, and is used to activate or block disjoint regions of the embedding space, as illustrated in Figure 5.2 (B).

Given a specific similarity dimension s, training triplets are defined as $T_s = \{t_s^i\}_{i=1}^{N}$, with each triplet given by:

$$t_s^i = (x_a^i, x_p^i, x_n^i; s), \tag{5.2}$$

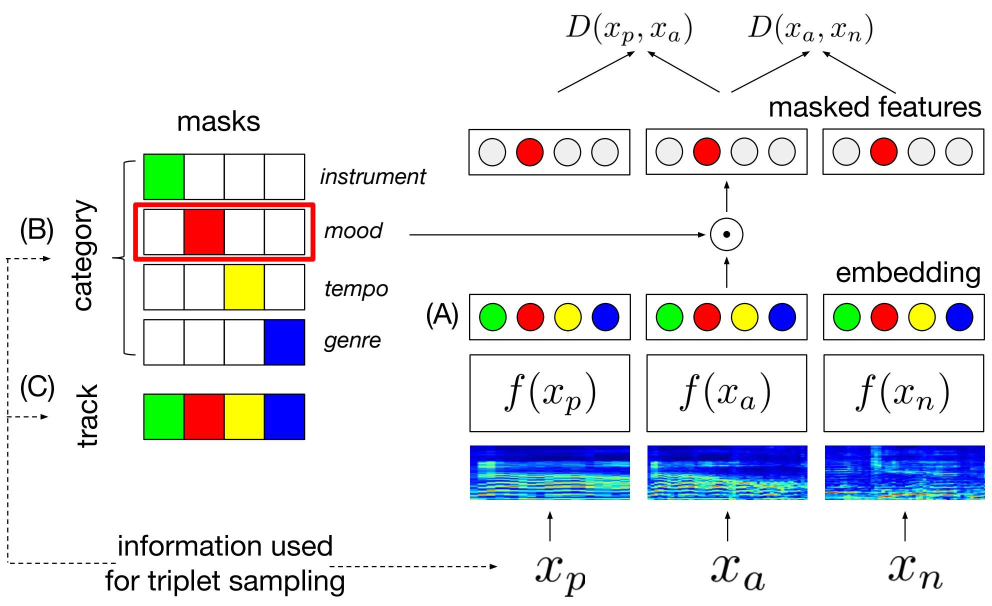


Figure 5.2: Our proposed approach. (A) Standard triplet-based deep metric model, (B) conditional similarity masking, and (C) track regularization.

and the training set combining triplets sampled from all similarity dimensions is defined as $T_S = \{T_s\}_{s=1}^{S}$. Consequently, we update the distance function to:

$$D(x_i, x_j; s) = ||f(x_i) \circ m_s - f(x_j) \circ m_s||_2, \tag{5.3}$$

such that the mask m_s only passes through the subspace of embedding features corresponding to similarity dimension s during training and $\circ$ denotes Hadamard product. Accordingly, the loss is updated to:

$$L(t_s) = \max\{0, D(x_a, x_p; m_s) - D(x_a, x_n; m_s) + \Delta\}. \tag{5.4}$$

Track regularization

As noted earlier, music replacement requires retrieved songs to sound as close as possible to the query example. To this end, we propose to complement the aforementioned multidimensional metric learning approach with a regularization technique we refer to as "track regularization". The approach involves sampling an additional set of triplets solely based on the track (song) information: the anchor and positive are both sampled from the same song, while the negative is sampled from a different song. While this sampling was used previously to learn high-specificity music similarity directly [88], here we use it as a "similarity regularization" technique to enforce a certain degree of consistency across the entire (multidimensional) embedding space. With this regularization, our final loss is given by:

$$L(t_c, t_t) = L(t_c) + \lambda L(t_t), \tag{5.5}$$

where t_c are all triplets sampled from the various music similarity dimensions corresponding to disjoint sub-embedding spaces, t_t are triplets sampled using track information, and λ allows us to control the trade-off between semantic similarity (low-specificity) and overall track-based similarity (high specificity). Importantly, for track-based triplets, we use a mask with a value of one for all feature dimensions, meaning the regularization is applied to the complete embedding space to capture track similarity across all musical dimensions. Alternatively, this can be thought of as not applying any masking on the embedding space.

5.1.3 Experimental Design

Dataset and input features

For our experiments, we use the Million Song Dataset (MSD) [36]. Based on preliminary user studies on music replacement, we identify four relevant musical dimensions to consider: *genre*, *mood*, *instrumentation*, and *tempo*. To determine whether two songs are similar along these dimensions, we use Last.FM tag annotations associated with MSD tracks which have been previously grouped into different categories [31], resulting in 28 genre tags, 12 mood tags, and 5 instrument tags. Since the annotations lack tempo tags, we extract an algorithmic tempo estimate per track using the Madmom Python library [90, 91]. Two tracks are considered similar along a certain musical dimension (genre, mood, instruments) if they share at least one tag in that category, or are within 5 BPM of each other in the case of tempo. For track-based triplets, we ensure there is no more than 50% overlap between the anchor and positive samples. We split the data following [27], giving 201680, 11774, and 28435 samples for the train, validation, and test sets, respectively.

For training, we use 3-second excerpts represented as a log-scaled mel-spectrogram S, extracted with librosa [92]. We use a window size of 23 ms with 50% overlap and compute 128 mel-bands per frame with the following log-compression: $log_{10}(1+10*S)$, resulting in input dimensions of 129×128 as in [82]. The representation is z-score standardized using fixed mean and standard deviation values of 0.2 and 0.25, respectively.

Model architecture and training parameters

For choosing the triplet network architecture, we ran preliminary experiments with several state-of-the-art convolutional building blocks [93], including a basic conv-batchnorm-maxpool block, ResNet [57], Squeeze-and-Excitation [94], and Inception [95]. Having identified the Inception block as the best option, we use the following model architecture: we start with 64 convolutional filters with a 5×5 kernel followed by 2×2 strided max-pooling, followed by six Inception blocks each comprising a "naïve" inception module with stride 2 followed by another inception module with a final output dimensionality of 256 [95]. We use ReLU nonlinearities for all layers, and apply $L2$ normalization to the embedding features prior to computing the distance [87].

Since our total embedding size is 256 and we consider four music similarity dimensions (genre, mood, instruments, tempo), each with a disjoint subspace of size 64. We also experimented with a trainable masking layer [11] (as opposed to fixed disjoint masks), but found it did not lead to any significant improvement. Moreover, using fixed masks has the added benefit of allowing us to weight each musical dimension independently post-hoc which, as noted earlier, is a desirable user interaction paradigm. We use the Adam optimizer [96] for training. We initialize the learning rate to 0.01 and reduce it by a factor of 5 when the validation loss does not decrease for 4 epochs, up to 5 times, after which we apply early stopping. The margin for the triplet loss is set to 0.1. And, after empirically hearing the properties of similarity space, λ was set to 0.5 when track regularization is applied.

Used space	Embedding Features	Genre	Mood	Instruments	Tempo	Overall
All-dimensions	MFCC-VQ	0.563	0.481	0.495	0.516	0.514
	Track	0.611	0.595	0.531	0.534	0.568
	Category	0.647	0.633	0.562	0.875	0.679
	Category + track regularization	0.647	0.627	0.561	0.891	0.681
	Category + disentanglement	0.708	0.717	0.657	0.783	0.716
	Category + disentanglement + track regularization	0.693	0.704	0.626	0.836	0.715
Sub-dimensions	Set of specialized networks	0.708	0.619	0.603	0.942	0.718
	Category + disentanglement	0.785	0.790	0.798	0.955	0.832
	Category + disentanglement + track regularization	0.765	0.743	0.700	0.953	0.790

5mm

Table 5.1: Prediction accuracy of category-based (genre, mood, instruments, tempo) triplets.

Embedding Features	Track	User
MFCC-VQ	0.833	0.654
Track	0.950	0.763
Category	0.975	0.766
Category + track regularization	0.980	0.740
Category + disentanglement	0.985	0.763
Category + disentanglement + track regularization	0.988	0.792

Table 5.2: Results on track-based and user-based triplets.

Evaluation metrics and user-study

For evaluation we use a set of held-out triplets sampled from the test set. We sample 40,000 triplets per music dimension (genre, mood, instruments, tempo) as well as 40,000 triplets based on track information. To simulate our application scenario, we use triplets of full songs for evaluation, the only exception being track-based triplets, where we stick to 3 second excerpts since the anchor and positive are sampled from the same song and should not overlap by more than 50%. The embedding for a full song is obtained by computing embedding frames from 3-second non-overlapping windows and averaging them over the time dimension. Given a test triplet, a model is evaluated by testing whether the embedding distance between the anchor and positive samples is smaller than the distance between the anchor and negative (score of 1), or greater (score of 0). The scores for all triplets are averaged to obtain a final score between 0 (worst) and 1 (best).

To determine whether human subjects concur with the above quantitative evaluation, we also randomly sampled 4,000 triplets from the test set and asked people to annotate which track sounded more similar to the anchor (positive or negative) without showing which was which. Each triplet was annotated by 5-12 people, resulting in 39,440 human annotations. We then calculated the annotator agreement per triplet, defined as the ratio between the majority vote and total number of annotations, and filtered out triplets where the agreement was below 0.9, resulting in 879 high-agreement human-annotated triplets. Since similarity judgements have a high degree of subjectivity, in this way, we can limit the scope of our human evaluation to triplets where there is broad annotator agreement. Models are evaluated against these triplets as described earlier, obtaining a score between 0–1 in terms of consistency with user ratings.

Baseline method

As a strong baseline, we implement a vector quantization method that has been used for both similarity-based music retrieval and auto-tagging [9, 97]. We compute 13 MFCC coefficients and their first and second derivatives per frame for each track, randomly select 2,500,000 frames from all tracks and cluster them using K-means with $K = 1024$ to produce a dictionary [9]. Given the dictionary, a track embedding is obtained by assigning each MFCC frame to its closest cluster and computing a normalized histogram of cluster assignments. The distance between any two tracks is then given by the Euclidean distance between their normalized histograms [9].

5.1.4 Results

In Table 5.1, we present the numerical results obtained for each of the four held-out triplet sets corresponding to a music similarity dimension, as well as aggregated scores over all four triplet sets ("Overall"). The "Used space" column indicates which subset of the embedding space was used to compute the distance between pairs of tracks, where "all dimensions" means all embedding features were used ($f(x)$), whereas "sub-dimensions" means only the subspace corresponding to the musical dimension ($f(x) \circ m_s$) from which the test triplets were sampled was used. We compare six models plus the baseline, specified in the "embedding features" column. The "Track" model was trained on triplets sampled based on track-information only, the "Category" model was trained on triplets sampled from the four music similarity dimensions (categories) of genre, mood, instruments and tempo, including both with and without disentanglement (subspace masking) and track regularization. For disentangled models, we include an additional baseline, "Set of specialized networks", which is comprised of four separate triplet-loss networks, each trained exclusively on triplets sampled from one of the four musical dimensions.

We see that all deep metric learning models outperform the MFCC-VQ baseline. More importantly, disentangling the embedding improves performance in almost all cases, with our disentangled model trained on all triplets jointly (Category + disentanglement) even outperforming the specialized networks trained separately on each dimension.

As one might expect, track regularization decreases numerical performance on each of the four triplet test sets, as it enforces all embedding subspaces to respect a global notion of track similarity. The key question is how does it affect model performance when compared against the human ratings obtained from our user study, presented in Table 5.2. As a sanity check, we start by evaluating our models against the track-based triplet test-set, presented in the "Track" column. We see that, as expected, track-regularization increases performance on this high-specificity set. Somewhat surprisingly, training on category-sampled triplets outperforms training on track-sampled triplets, with disentanglement increasing performance further. Next, we turn to the results obtained from the user study, presented in the "User" column. We see that our proposed approach outperforms the baseline, and, as per our initial hypothesis, track regularization increases the overall user agreement with our model's similarity ratings when training on category triplets with disentanglement.

5.1.5 Contribution Summary

In this paper, we introduce a novel approach for deep metric learning of a disentangled, *multidimensional*, music similarity space. We use Conditional Similarity Networks trained on a combination of user tags and algorithmic estimates, and introduce track regularization to control for retrieval specificity. Through a series of experiments, including both a quantitative evaluation and a user study, we demonstrate that our proposed approach outperforms several baselines, with per-dimension similarity performance increasing due to the disentangling of the embedding space, and agreement with human annotations increasing as a result of track regularization. Our solution is particularly relevant to the music replacement problem, and opens the door to novel interaction paradigms which permit the user to select which music dimensions they care about for retrieval, how to weight their relative importance, and how to balance subspace similarity versus high-specificity overall similarity. This approach can further be extended to general audio similarity such as voice similarity based on their speaker's condition, phonation, or prosody. In the future, we plan to conduct further user studies to determine human agreement when considering each musical dimension in isolation, and evaluate the performance of our model against these ratings. We also plan to explore and evaluate our proposed approach for multi-query retrieval (query-by-multiple-examples) and mix-and-match scenarios where the user is interested in finding songs whose characteristics match the subspaces of different songs (e.g. the genre of example A with the tempo of example B).

5.2 Metric Learning vs Classification for Disentangled Music Representation Learning

Deep representation learning offers a powerful paradigm for mapping input data onto an organized embedding space and is useful for many music information retrieval tasks. Two central methods for representation learning include deep metric learning and classification, both having the same goal of learning a representation that can generalize well across tasks. Along with generalization, the emerging concept of disentangled representations is also of great interest, where multiple semantic concepts (e.g., genre, mood, instrumentation) are learned jointly but remain separable in the learned representation space. In this paper we present a single representation learning framework that elucidates the relationship between metric learning, classification, and disentanglement in a holistic manner. For this, we (1) outline past work on the relationship between metric learning and classification, (2) extend this relationship to multi-label data by exploring three different learning approaches and their disentangled versions, and (3) evaluate all models on four tasks (training time, similarity retrieval, auto-tagging, and triplet prediction). We find that classification-based models are generally advantageous for training time, similarity retrieval, and auto-tagging, while deep metric learning exhibits better performance for triplet-prediction. Finally, we show that our proposed approach yields state-of-the-art results for music auto-tagging.

5.2.1 Problem

Learning a good representation, or embedding space, is a key goal in deep learning and is central to music classification and retrieval tasks. An important quality of a good representation is its generalization capability, i.e., its applicability to a diverse set of downstream tasks, including those relying on small datasets in a transfer learning setting [73, 82, 98]. While numerous representation learning methods have been explored to date, two learning paradigms are particularly common: deep metric learning and classification-based representation learning. The former is based on deriving similarity scores (or distances) between examples, while the latter is achieved via a cross-entropy loss over similarity scores between example and class centroids.

While both paradigms share the goal of learning a generalizable representation, the results from each approach are generally different. For example, a learned representation optimized via a classification task may perform poorly on a similarity-search task, and vice versa. While recent studies have elucidated the theoretical relationships between these paradigms and validated them through experimental findings [99], these developments have not been explored in the music domain. Furthermore, the relationship has not been explored for multi-label data, which is central to many music information retrieval tasks.

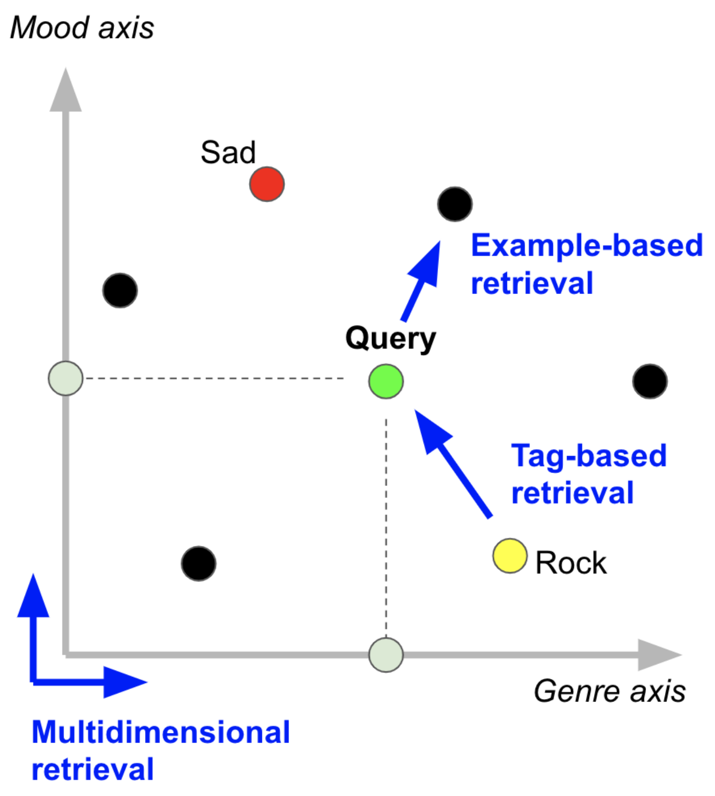


Figure 5.3: A disentangled music representation space. The green dot depicts a query song, the black dots depict retrieval songs, the red and yellow dots depict centroids of musical concepts, the gray arrows depict multidimensional axis, and the blue arrows depict retrieval methods.

Beyond seeking a representation that generalizes across tasks, the emerging concept of disentangled representations [100, 101] is of great interest for music applications. Music is often labeled with multiple semantic dimensions simultaneously (e.g., genre, mood, and instrumentation) and learning a representation that can capture this structure is advantageous. We often need to search for music that is similar along a particular semantic dimension in one application (e.g., a music playlist with lighthearted mood), while requiring music similar along a different semantic dimension for another application (e.g., era for musicological analysis). Disentangled representations allow us to address both problems with a single model, and were recently proposed for audio-based music similarity search [102]. However, this study only explored disentanglement via a single deep metric learning approach, and the applicability and performance of more recent metric- and classification-based learning methods is yet to be explored.

In this paper, we present a unified representation learning framework that elucidates the relationship between metric learning, classification, and disentanglement. First, we outline past work on the relationship between metric learning and classification. We then extend this relationship to multi-label and multi-concept data (common to music applications) by exploring three different learning approaches and their disentangled versions – two of which are novel to this work. Finally, we evaluate all models against four tasks (training time, similarity retrieval, auto-tagging, and triplet prediction) and compare various aspects of the learned representations.

5.2.2 Related Work

Metric Learning and Classification

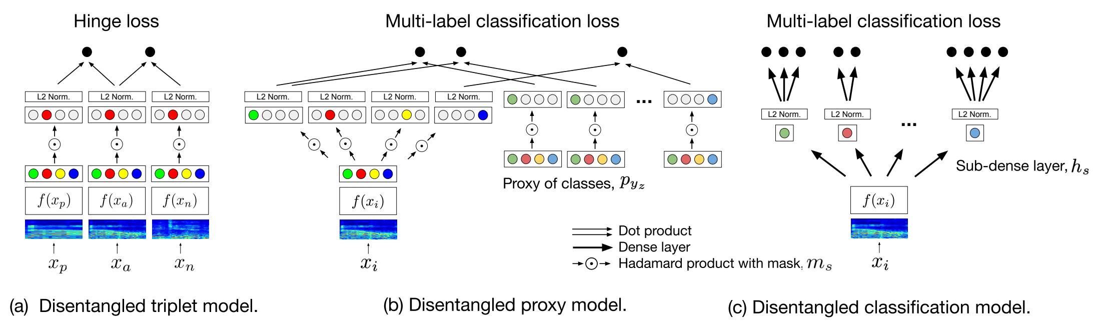
The goal of distance metric learning is to obtain an embedding space where similar items are close together and dissimilar items are far apart. A common strategy is to use pairwise [103, 104] or triplet-based samples to train a model [86, 105–107]. An important advantage of deep metric learning is that it can efficiently model an extremely large number of classes (e.g., for face recognition) [107]. However, training models using this strategy are relatively slow as models operate on triplets of input samples [108]. Recently, more efficient sampling techniques have been proposed to speed up convergence, including hard negative mining, semi-hard negative mining [107], distance weighted sampling [109], and proxy-based training [108]. Proxy-based training [108] assigns one or several proxies to each class (given by per-class embedding centroids) and optimizes the learned space by comparing embedded input samples to proxies instead of directly comparing them to positive and negative samples. This reduces training time significantly while improving retrieval performance on images.

Classification models, on the other hand, are typically trained such that classes are linearly separable in the embedding space of the last hidden layer of the deep neural network. Since classification models are not optimized based on distances in the learned embedding space, they may not perform well when directly used for similarity-based retrieval. To overcome this, recent work proposed the application of a normalization layer over the embedding space during training, and showed that this simple technique increases model performance on similarity-based image retrieval [99].

Recent and parallel advances in both paradigms (metric- and classification-based learning) have shown that there is an inherent link between them [99, 110, 111]. The per-class embedding centroids used in proxy-based training are, in fact, equivalent to the per-class vectors obtained from the linear transformation in the last hidden layer of a classification model [110]. Further, a recent comparative study demonstrated that the loss function of a triplet-based model is equivalent to that of a classification model up to a smoothing factor for single-label, multi-class data [110]. These findings suggest that deep metric- and classification-based learning are not as different as initially thought and we could, potentially, use either to learn a representation that generalizes well to both similarity-based retrieval and classification tasks.

Disentangled Representation Learning

Another important measure of representation learning is *disentanglement* [112]. Recently, Lee et al. adapted Conditional Similarity Networks (CSN) applied to triplet-based deep metric learning to the music domain [11, 102]. The main idea in CSN is to apply a masking function over the embedding space, where each mask corresponds to a different semantic dimension of similarity corresponding to musical notions such as genre, mood, instrument and tempo. They showed that the disentangled music representation not only enables multidimensional music search via its sub-dimensions, but also improves general music retrieval performance when all embedding dimensions are used. However, CSN for disentangled music representation learning was only explored using a deep metric learning strategy, and classification-based approaches were not studied. Considering the close relationship between the two, we propose to study disentanglement under classification, particularly for multi-labeled music data, and compare and contrast it to disentanglement via metric learning.



Hinge loss
Multi-label classification loss
Multi-label classification loss
L2 Norm.
$f(x_p)$
$f(x_a)$
$f(x_n)$
x_p
x_a
x_n
$f(x_i)$
x_i
Proxy of classes, p_{y_z}
Sub-dense layer, h_s
Dot product
Dense layer
Hadamard product with mask, m_s
(a) Disentangled triplet model.
(b) Disentangled proxy model.
(c) Disentangled classification model.

Figure 5.4: A unified framework for disentangled triplet- and proxy-based metric learning and multi-label classification.

5.2.3 Disentangled Learning Models

In this section, we introduce three disentangled learning methods, which are triplet-based, proxy-based, and classification-based models. The first model was previously developed [102], and the latter two are novel contributions. The overall architectures are illustrated in Figure 5.4. In the following descriptions, x denotes a data point, $f(\cdot)$ a nonlinear embedding function, y a multi-hot class label, and s a category (or a similarity notion such as mood, genre or instrumentation) of y. For example, if y_z is *rock*, then s_{y_z} is *genre*.

Triplet-based Model

Disentangled triplet-based models were recently proposed in [11, 102]. We first define a triplet as $t = (x_a, x_p, x_n; y_z)$, where x_a is the anchor sample, x_p is the positive sample, and x_n is the negative sample. x_a and x_p are sampled to have the same positive label y_z, while x_n is negative for y_z. Then, the basic triplet loss is defined as

$$L(t) = \max\{0, D(f(x_a), f(x_n)) - D(f(x_a), f(x_p)) + \Delta\}, \tag{5.6}$$

where $D(f(x_i), f(x_j)) = cos(f(x_i), f(x_j))$ is a distance metric, and Δ is a margin value [106]. To disentangle the embedding feature of size d, a masking function $m_s \in \mathbb{R}^d$ is applied. The number of masks corresponds to the number of similarity notions s and each mask occupies certain dimensions of the $\mathbb{R}^d$ space evenly as illustrated in Figure 5.4 (a). Thus, when the $t = (x_a, x_p, x_n; y_z)$ is used, a mask for the similarity notion s_{y_z} is applied to the embedding feature space. The loss for training the model is given by:

$$\begin{aligned} L(t) = \max\{0, &D(f(x_a) \circ m_s, f(x_n) \circ m_s) \\ &-D(f(x_a) \circ m_s, f(x_p) \circ m_s) + \Delta\}, \end{aligned} \tag{5.7}$$

where $\circ$ denotes the Hadamard product.

Proxy-based Model

The core idea of proxy-based metric learning is that proxy embeddings are learned and assigned to each class and used to measure the distance to an anchor data point instead of directly measuring distances to pairs or triplet data samples [108]. This can be interpreted as a supervised clustering algorithm, where proxies play a role of class centroids. In this approach, the distance metric becomes

$$D(f(x_i), p_{y_z}) = cos(f(x_i), p_{y_z}) = \frac{f(x_i)}{||f(x_i)||} \cdot \frac{p_{y_z}}{||p_{y_z}||}, \tag{5.8}$$

where x_i is a data point, p_{y_z} is a proxy for class y_z, and $\cdot$ is the dot product. If the data is single-labeled (multi-class), one can apply triplet loss, Neighborhood Component Analysis (NCA) loss [113], or Softmax loss over the above distance metric [108,110], but with our multi-labeled data, it is not directly applicable. To address this, we replace these losses with a multi-label classification loss, i.e., binary cross entropy. The prediction score for each class becomes

$$\hat{y_z} = sigmoid(D(f(x_i), p_{y_z})), \tag{5.9}$$

and the loss is

$$L(x_i) = \sum_z [-y_z log(\hat{y_z}) - (1 - y_z) log(1 - \hat{y_z})]. \tag{5.10}$$

However, from our preliminary experiments, we found that the sigmoid function with cosine similarity score causes numerical problem in optimization. We speculate that the reason for this is that the cosine similarity score (bounded between -1 to +1) only activates the linear regions of the downstream sigmoid activation, reducing model capacity.[1] Therefore, we modify the distance metric to be

$$D(f(x_i), p_{y_z}) = \frac{f(x_i)}{||f(x_i)||} \cdot p_{y_z}, \tag{5.11}$$

to ensure that both the learned embedding space is normalized and the sigmoid activations can have nonlinear properties.

From this basic multi-label proxy-based model, we expand the model by applying the masking function as used in the disentangled triplet-based model. Then, the prediction score for each class is updated to

$$\hat{y_z} = sigmoid(D(f(x_i) \circ m_s, p_{y_z} \circ m_s)), \tag{5.12}$$

as illustrated in Figure 5.4 (b).

Classification-based Model

Classification-based metric learning has recently been explored [99, 110]. The core idea is to apply a normalization layer on the embedding feature space. This simple technique ensures that the learned representation has unit length and makes similarity-based retrieval more effective compared to the vanilla classification model. Therefore, the prediction score of classification-based metric learning model for each class is

$$\hat{y_z} = sigmoid(\frac{f(x_i)}{||f(x_i)||} \cdot c_{y_z}), \tag{5.13}$$

where c_{y_z} is a centroid for each class (parameters of the last hidden layer).[2] At this stage, we observe that the distance metric inside the sigmoid function of Equation 5.13 is equivalent to that of our modified distance metric in Equation 5.11 of the proxy-based model.

As for triplet-based metric learning, we extend classification-based metric learning to learn a disentangled embedding space. We begin from the disentangled distance

[1] In proxy-triplet loss, this type of numerical problem does not occur because they are relative comparison based losses. In proxy-NCA or proxy-Softmax loss, some of the previous works encountered similar problem, and solved the problem by applying a smoothing factor over the similarity score [99, 110, 114]. We also tested applying a smoothing factor, but for our multi-label classification problem, it turns out that the proposed modified distance metric is more effective.

[2] In our preliminary experiments, we found that removing the bias term does not decrease the model performance, so we did not include it in Equation 5.13.

metric, which is

$$\begin{aligned} D(f(x_i) \circ m_s, c_{y_z} \circ m_s) &= \frac{f(x_i) \circ m_s}{||f(x_i) \circ m_s||} \cdot (c_{y_z} \circ m_s) \\ &= \frac{1}{||f(x_i) \circ m_s||} \cdot (f(x_i) \circ m_s) \cdot (m_s \circ c_{y_z}). \end{aligned} \tag{5.14}$$

From the above equation, if we split $f(x_i)$ into the nonlinear function $f_{n-1}(x_i)$ and the embedding feature layer h (here, h layer includes nonlinear activation), then the equation becomes

$$\begin{aligned} &= \frac{1}{||f(x_i) \circ m_s||} \cdot (f_{n-1}(x_i) \cdot h \circ m_s) \cdot (m_s \circ c_{y_z}) \\ &= \frac{1}{||f(x_i) \circ m_s||} \cdot f_{n-1}(x_i) \cdot h \circ m_s \cdot m_s \circ c_{y_z}. \end{aligned} \tag{5.15}$$

In this equation, $(h \circ m_s \cdot m_s \circ)$ is actually a sub-dense layer that has the same dimensionality as the disjoint mask m_s, which is applied when $y_z \in s$. Henceforth, we denote the sub-dense layer h_s. Now, $||f(x_i) \circ m_s||$ can be replaced to $||f_{n-1}(x_i) \cdot h_s||$. Finally, the disentangled distance metric becomes

$$= \frac{1}{||f_{n-1}(x_i) \cdot h_s||} \cdot (f_{n-1}(x_i) \cdot h_s) \cdot c_{y_z}. \tag{5.16}$$

This is the same formula for multi-task learning in the multi-label classification problem formulation, surprisingly, proving a previously unknown link between the two concepts. We illustrate this disentangled classification-based model in Figure 5.4 (c). Through experimental evaluation, we further verify that this multi-task learning-based classification model is equivalent to the disentangled proxy-based model while being much simpler to implement and benchmark.

5.2.4 Experiments

Dataset and Input Features

For our experiments, we use the Million Song Dataset (MSD) [36] and Last.FM tag annotations associated with MSD tracks, which have been previously grouped into different categories [31], resulting in 28 genre tags, 12 mood tags, 5 instrument tags, and 5 era tags. We treat each category as a similarity notion s. We use these tags for evaluating similarity-based retrieval, auto-tagging, and triplet prediction tasks. The data are split into 201680, 11774, and 28435 samples for the train, validation, and test sets, respectively, following a previous auto-tagging benchmark [27]. For triplet prediction evaluation, we follow the same procedure as in [102], albeit switch one similarity notion (era replaces tempo) to match auto-tagging benchmarks. We sample 40,000 triplets per each similarity notion (genre, mood, instruments, era, track) and use a cleaned version of the *dim-sim* dataset to evaluate the models on human-annotated triplets.

The input to the embedding function $f(\cdot)$ is 3-second excerpts represented as a log-scaled mel-spectrogram S, extracted with librosa [92]. We use a window size of 23 ms with 50% overlap and 128 mel-bands, resulting in input dimensions of 129×128 as in [102]. The input features are z-scored standardized using fixed mean and standard deviation values of 0.2 and 0.25, respectively.

Backbone Model and Training Parameters

For the embedding function or backbone model $f(\cdot)$, we use the same architecture as described in [102], which is an Inception-based model [95]. The model is comprised of a convolution layer with 5×5 sized 64 filters followed by 2×2 strided max-pooling, followed by six Inception blocks. Each Inception block consist of two Inception modules, a *naïve* module and *dimension reduction* module, which are applied in sequence. Both of the modules include filters of mixed size, but the *naïve* module has 2×2 strides in the last convolution layers of the module, so that the spatial feature map is reduced, and the *dimension reduction* module has a fixed number of filters in the last convolution layers of the module, so that the feature map is fixed to 256 in the intermediate layers. At the end, one fully connected layer with 256 units is added, except for the disentangled (multi-task learning) classification-based model, which uses sub-dense layers instead of a single fully connected layer. We use ReLU nonlinearities for all layers.

Since our embedding dimensionality is 256 and we consider four music similarity notions (genre, mood, instruments, era), each has a disjoint subspace of size 64. For the disentangled (multi-task learning) classification-based model, the sub-dense layers are also 64 units each. We use the Adam optimizer [96] for training. We initialize the learning rate to 0.005 and reduce it by a factor of 5 when the validation loss does not decrease for 10 epochs, up to 5 times, after which we apply early stopping. The margin for the triplet-based models is set to 0.1.

Models	Normalization	Disentanglement	Training time	Similarity-based retrieval				Auto-tagging
			ratio	R@1	R@2	R@4	R@8	AUC
Triplet			1.87	31.8	45.2	59.9	73.0	0.815
Triplet			2.37	36.5	50.5	64.1	76.0	0.825
Triplet + track reg.			3.05	33.9	47.5	61.9	74.3	0.813
Proxy			1.11	**45.0**	**58.5**	**71.0**	**80.9**	**0.890**
Proxy			1.29	44.7	58.2	70.7	80.6	**0.890**
Classification			**1.00**	6.1	11.5	21.1	35.9	0.887
Classification			**1.00**	43.8	57.8	70.3	80.3	0.887
Classification			1.27	44.7	58.4	70.7	**80.9**	**0.890**

Table 5.3: Results for training time, similarity search, and auto-tagging.

Model	AUC
CRNN [31]	0.850
Self-attention [115]	0.881
Sample-level ReSE-2 [93]	0.885
Multi-level & multi-scale [27]	0.888
Proposed Model	**0.890**

Table 5.4: Auto-tagging SOTA comparison.

Evaluation Tasks

Our learned representations can be utilized for many applications, so there are many aspects to consider when evaluating representation learning models. Therefore, as a unified evaluation framework, we evaluate the models on four tasks: training time, similarity-based retrieval, auto-tagging, and triplet prediction.

Training Time

We first measure the overall training time to see the efficiency of the representation learning model. The training time is calculated as the total number of epochs multiplied by the time consumption of 1 epoch. Then, we report the value as a ratio to the shortest training time.

Similarity-based Retrieval

For the similarity-based retrieval evaluation, we use the recall@K (R@K) metric to measure retrieval quality following the standard evaluation setting in image retrieval [99,108–110,116]. This metric is useful for evaluating a search system because it measures the quality of the top K retrieved results, which are more important than long-tail retrieved results. The definition of the standard recall@K that is used for single-label problems is as follows. A query song is used to search a test set of recordings and retrieve similar sounding results. If one of the top K retrieved results has the same class label as the query song, the recall@K is set to 1, otherwise it is set to 0. This process is repeated for all samples in the test set and then averaged.

Our data is multi-labeled, however, so we adapt the standard single-label (multi-class) R@K metric to create a multi-label variant. Our definition is

$$R@K = \frac{1}{N}\sum_{q=1}^{N}\frac{n(y^q \cap (\cup_{i=1}^{K} y^i))}{n(y^q)}, \tag{5.17}$$

where N is the number of test samples, y^q is the ground truth labels of a query, and y^i is the ground truth labels of the top K retrieved results. And, $n(\cdot)$ denotes the number of the elements of a set. In this setup, if the set of labels of the top K retrieved results contains all the multiple labels of the query song, the recall@K is set to 1, otherwise it is set to the correct answer ratio. We report R@K when K is 1, 2, 4, and 8.

Auto-tagging

Music auto-tagging has been extensively studied in the literature with diverse model architectures [98]. As such, we follow standard benchmarking and evaluation criteria, and report area under the receiver-operator curve (AUC) to measure tag-based retrieval performance.

Unlike the proxy-based and classification-based approaches, the triplet-based model doesn't directly predict a class (or several classes) for a given input. Thus, we use the concept of prototypes to obtain classification result from the triplet-based models [13]. We first average all the embedding features of the training samples that are assigned to each class label to construct prototype (or centroid) of each class label. Then, we measure a distance between these prototypes and embedding feature of each sample and regard it as a prediction score for classification, which itself is directly used for AUC evaluation.

Embedding space	Models	Normalization	Disentanglement	Genre	Mood	Instruments	Era	Overall
Complete space	Triplet			0.771	0.725	0.653	0.701	0.712
	Triplet			0.762	0.744	0.696	0.733	0.733
	Triplet + track reg.			0.757	0.733	0.673	0.715	0.720
	Proxy			0.774	0.742	0.645	0.693	0.714
	Proxy			0.762	0.742	0.660	0.716	0.720
	Classification			0.783	0.745	0.659	0.723	0.728
	Classification			0.776	0.747	0.647	0.704	0.719
	Classification			0.758	0.742	0.659	0.715	0.719
Sub-space	Triplet			**0.790**	**0.785**	**0.798**	**0.797**	**0.792**
	Triplet track reg.			0.775	0.748	0.743	0.742	0.752
	Proxy			0.777	0.740	0.734	0.700	0.738
	Classification			0.775	0.739	0.732	0.701	0.737

Table 5.5: Results on tag-based triplets.

Models	Normalization	Disentanglement	Track	Human-labeled
Triplet			0.957	0.820
Triplet			0.964	0.820
Triplet + track reg.			0.961	**0.852**
Proxy			0.978	0.784
Proxy			0.978	0.791
Classification			0.978	0.780
Classification			0.978	0.795
Classification			**0.984**	0.801

Table 5.6: Results on track-based & human-labeled triplets.

Triplet Prediction

Triplet prediction score is simply measured by counting the number of correct predictions among all test triplets. Here, it is regarded as correct if the distance between the embedding features of the anchor and the positive is smaller than that of the distance between the anchor and the negative.

5.2.5 Results

In Table 5.3, we present the results for training time, similarity-based retrieval, and auto-tagging. We compare a total of eight models, which are categorized into three learning methods: triplet-based, proxy-based, and classification-based models. "Disentanglement" indicates whether a CSN masking function is applied to each learning method, and "Normalization" indicates whether a normalization layer is applied to the model's embedding layer. "Track regularization" (track reg.) indicates whether, in addition to tag-based triplets, we also sample triplets by taking the anchor and positive from the same track and the negative from a different track, as proposed in [102].

First, we see that the training time, represented as the ratio between each model's training time and the training time of the fastest approach, is significantly reduced for the proxy-based and classification-based models compared to the triplet-based models. This is because each training sample for the triplet model is actually composed of 3 inputs (anchor, positive and negative) or even 5 when track regularization is also applied, whereas the proxy-based and classification-based approaches only require one input per training sample.

Second, for similarity-based retrieval, we see that the vanilla classification model without a normalization layer exhibits poor performance. This confirms our conjecture that using the representation learned by the classification model without normalization layer directly is not optimal for similarity-based retrieval, as the model is not optimized based on distances in the learned embedding space. We also see that the proxy- and classification-based models are superior to the triplet-based models across the board. We hypothesize that this is due to the latter strategy using only a single label per training sample, whereas the former two use all (multi-)labels for each training sample, thus exploiting a richer signal during training.

Third, for auto-tagging, we see that the proxy-based and classification-based models outperform the triplet-based model by a large margin. As expected, the vanilla classification-based model performs well on this task. In Table 5.4, we compare our proposed classification-based disentangled model to the state of the art (SOTA) for music auto-tagging. Our model outperforms all baselines, setting the new state-of-the-art for music auto-tagging.

Fourth, for triplet prediction, we report tag-based triplet results in Table 5.5 using different similarity dimensions (genre, mood, instruments, era), and in Table 5.6 the results for track-based and human-labeled triplets. The "Embedding space" column indicates whether we use the complete embedding space to measure the similarity between pairs of examples, or whether we only use the disjoint sub-space ($f(x_i) \cdot m_s$ or h_s) corresponding to the similarity notion s used to sample the test triplets (genre, mood, instruments or era). In Table 5.6 we use the complete space.

Fifth, in Table 5.5 we see that while proxy- and classification-based embeddings are superior for music retrieval and tagging, triplet-based embeddings perform better (unsurprisingly) on the triplet-prediction task. It is noteworthy that while the triplet task is often used as a proxy for evaluating music similarity modelling, models that do best on this task are not necessarily the best at downstream retrieval tasks as evidenced by Table 5.3. In Table 5.6, we also see that while classification-based embeddings perform better at predicting track-based triplet similarity, triplet-based embeddings perform better when it comes to matching human judgements of triplet similarity. This is particularly true when we apply triplet learning with track regularization, in accordance with previous work [102].

5.2.6 Visualization of Disentangled Space

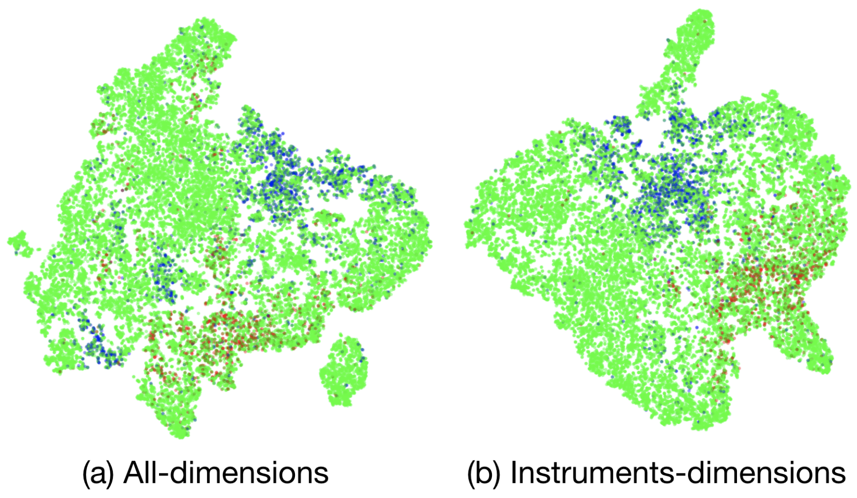


Figure 5.5: t-SNE plot of test set embedding features. The blue dots are labeled positive for the female vocalists tag, the red dots are labeled positive for the instrumental tag, and the green dots are negative.

To qualitatively evaluate the disentangled representation space learned by our model, we visualize the embeddings of the test set as a t-SNE plot [117] in Figure 5.5. We take embeddings from the disentangled triplet model and highlight samples with the *female vocalists* and *instrumental* tags as an example. While the highlighted samples are relatively dispersed when considering all dimensions, we see that they are nicely clustered together when only considering the instrument sub-space of the embedding. This illustrates the benefits of a disentangled space, which supports both global similarity and specialized similarity over specific music dimensions.

5.2.7 Contribution Summary

In this paper, we presented a detailed study of metric-based and classification-based learning approaches for music representation learning. We extended both strategies to learn disentangled spaces from multi-label data, and showed both analytically and empirically that under certain conditions, proxy-based learning is equivalent to classification-based learning. We benchmark multiple variants of each strategy in terms of training efficiency and performance on music retrieval, auto-tagging, and triplet prediction tasks. Our results show that, when coupled with disentanglement and normalization, classification-based representation learning produces superior benchmark results on all tasks, except for triplet prediction where triplet models are (predictably) strong performers, indicating that triplet prediction is not necessarily a reliable proxy for real-world retrieval performance. Our best performing disentangled model obtains state-of-the-art results for music auto-tagging, outperforming all previous baselines. Finally, we complement our quantitative analysis with qualitative results that further illustrate the benefits of learning a disentangled music embedding space.

Chapter 6. Conclusions

In this chapter, I will provide a summary of research contributions and practical ways of utilizing similarity-based deep learning for music retrieval. In addition, future work will be introduced.

6.1 Summary

We have explored and presented various individual modules of content-based music retrieval system. A content-based music retrieval system is mainly composed of three modules which are an audio embedding model, groundtruth labels, and training methods. Throughout this thesis, we proposed new methods for each of module and further extended the conventional content-based music retrieval scenarios to multidimensional music retrieval case.

In chapter 3, we presented sample-level deep convolutional neural networks using raw waveforms. This model is useful in efficiency, when the data storage is limited, because the model directly takes raw waveforms compared to mel-spectrogram based approaches, and we found that downsampling music audio down to 8000 Hz does not significantly degrade performance but it saves training time. Also, we visualized the spectrum of the learned filters for each sampling rate and found that the SampleCNN model is actively focusing on (or zoom in on) important low-frequency bands.

In chapter 4, we investigated the usefulness of artist label as a groundtruth label to train similarity-based deep learning model. Traditionally, such metric learning methods are trained with semantic labels such as tags. In this work, we compared artist-label model and tag-label model, and verified that the artist-label model is as useful as tag-label model. This suggests that free and objective metadata such as artist label is as useful as expensive tag annotations.

In chapter 5, we investigated disentangled metric learning methods and opens an application of multidimensional music retrieval. The results show that tag-based metric learning model with track regularization, that is merging different specificity level of information helps in building general music similarity space. In addition, we connect the relationship between metric learning and classification. By doing so we verified that the traditional classification model with simple normalization layer, we can build even stronger similarity space than the conventional triplet-based metric learning approach.

Through the exploration of this thesis, I hope we will be able to develop a better content-based music search system.

6.2 Future Work

Future work will be largely in three directions: an advanced model methodology, active learning, and exploration of human music similarity.

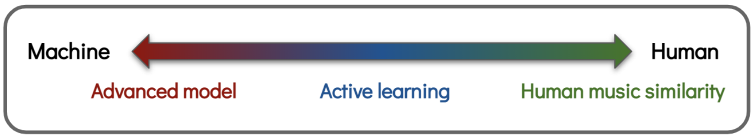


Figure 6.1: Future work.

First, an advanced model will be explored. We have mostly explored metric learning and classification for music search system using semantic tags and objective metadata of music. If more similarity notions that can be added in training phase, then the model can have more diverse characteristics of different similarity notions and if it is merged with multidimensional search application, we can search for music with more similarity notions. Therefore, we plan to add Music Information Retrieval (MIR) features as an additional similarity notion such as energy or the number of onsets.

Second, active learning for search will be explored. Search is not a one-shot experience, we search for items repeatedly until we find satisfactory results. And, active learning can provide adaptive search experience to each user. This is relatively an unexplored area in MIR community and I think we can compensate a disadvantage of content-based music search system compared to recommendation system by applying active learning methods, which is music search reflecting the characteristics of each user.

Third, deep understanding of human music similarity can give great insight into the development of similarity-based music search systems. There is not much work on large-scale, data-driven understanding of human music similarity in MIR community, and if we perform this study, and find several types of music similarity, then we can develop more satisfactory system. The insights gained from this kind of study will be of great help in setting future research direction.

Bibliography

[1] Defferrard, M., Benzi, K., Vandergheynst, P., and Bresson, X., “Fma: A dataset for music analysis.” in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 316–323, 2017.

[2] Casey, M., Veltkamp, R., Goto, M., Leman, M., Rhodes, C., and Slaney, M. (2008). “Content-based music information retrieval: Current directions and future challenges.” *Proc. of the IEEE*, 96.

[3] Celma, O., “Music recommendation.” in *Music recommendation and discovery*, Springer, 2010.

[4] Grosche, P., Müller, M., and Serrà, J., “Audio content-based music retrieval.” in *Dagstuhl Follow-Ups*, vol. 3, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2012.

[5] Nam, J., Herrera, J., Slaney, M., and Smith, J. O., “Learning sparse feature representations for music annotation and retrieval.” in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pp. 565–570, 2012.

[6] Dieleman, S., Brakel, P., and Schrauwen, B., “Audio-based music classification with a pretrained convolutional network.” in *Proc. of the International Society for Music Information Retrieval Conference*, pp. 669–674, 2011.

[7] Choi, K., Fazekas, G., and Sandler, M., “Automatic tagging using deep convolutional neural networks.” in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 805–811, 2016.

[8] Pons, J., Lidy, T., and Serra, X., “Experimenting with musically motivated convolutional neural networks.” in *Proc. of the International Workshop on Content-Based Multimedia Indexing (CBMI)*, pp. 1–6, 2016.

[9] McFee, B., Barrington, L., and Lanckriet, G. (2012). “Learning content similarity for music recommendation.” *TASLP*.

[10] Wolff, D., Stober, S., Nürnberger, A., and Weyde, T., “A systematic comparison of music similarity adaptation approaches.” in *ISMIR*, FEUP Edições, 2012.

[11] Veit, A., Belongie, S., and Karaletsos, T., "Conditional similarity networks." in *CVPR*.

[12] Dieleman, S., and Schrauwen, B., "End-to-end learning for music audio." in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6964–6968, 2014.

[13] Snell, J., Swersky, K., and Zemel, R., "Prototypical networks for few-shot learning." in *NeurIPS*, 2017.

[14] Krizhevsky, A., Sutskever, I., and Hinton, G. E., "Imagenet classification with deep convolutional neural networks." in *NIPS*, pp. 1097–1105, 2012.

[15] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J., "Distributed representations of words and phrases and their compositionality." in *Advances in neural information processing systems (NIPS)*, pp. 3111–3119, 2013.

[16] Zhang, X., Zhao, J., and LeCun, Y., "Character-level convolutional networks for text classification." in *NIPS*, pp. 649–657, 2015.

[17] Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2016). "Character-aware neural language models." *AAAI*.

[18] Sainath, T. N., Weiss, R. J., Senior, A. W., Wilson, K. W., and Vinyals, O., "Learning the speech front-end with raw waveform cldnns.." in *INTERSPEECH*, pp. 1–5, 2015.

[19] Collobert, R., Puhrsch, C., and Synnaeve, G. (2016). "Wav2letter: an end-to-end convnet-based speech recognition system." *arXiv preprint arXiv:1609.03193*.

[20] Zhu, Z., Engel, J. H., and Hannun, A., "Learning multiscale features directly from waveforms." in *INTERSPEECH*, 2016.

[21] Ardila, D., Resnick, C., Roberts, A., and Eck, D., "Audio deepdream: optimizing raw audio with convolutional networks." in *ISMIR, Late Breaking / Demo*, 2016.

[22] Thickstun, J., Harchaoui, Z., and Kakade, S. (2017). "Learning features of music from scratch." *ICLR*.

[23] Dai, W., Dai, C., Qu, S., Li, J., and Das, S., "Very deep convolutional neural networks for raw waveforms." in *ICASSP*, pp. 421–425, IEEE, 2017.

[24] Aytar, Y., Vondrick, C., and Torralba, A., "Soundnet: Learning sound representations from unlabeled video." in *NIPS*, pp. 892–900, 2016.

[25] Lee, J., Park, J., Kim, K. L., and Nam, J. (2017). "Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms." *Sound and Music Computing Conference (SMC)*, pp. 220–226.

[26] Simonyan, K., and Zisserman, A. (2015). "Very deep convolutional networks for large-scale image recognition." *ICLR*.

[27] Lee, J., and Nam, J. (2017). "Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging." *SPL*, 24.

[28] Tokozume, Y., and Harada, T., "Learning environmental sounds with end-to-end convolutional neural network." in *IEEE ICASSP*, pp. 2721–2725, 2017.

[29] Palaz, D., Doss, M. M., and Collobert, R., "Convolutional neural networks-based continuous speech recognition using raw speech signal." in *IEEE ICASSP*, pp. 4295–4299, 2015.

[30] Palaz, D., Collobert, R., et al., "Analysis of cnn-based speech recognition system using raw speech as input." tech. rep., Idiap, 2015.

[31] Choi, K., Fazekas, G., Sandler, M., and Cho, K., "Convolutional recurrent neural networks for music classification." in *ICASSP*, IEEE, 2017.

[32] Tzanetakis, G., and Cook, P. (2002). "Musical genre classification of audio signals." *IEEE Transactions on speech and audio processing*, 10, pp. 293–302.

[33] Kereliuk, C., Sturm, B. L., and Larsen, J. (2015). "Deep learning and music adversaries." *IEEE Transactions on Multimedia*, 17, pp. 2059–2071.

[34] Law, E., West, K., Mandel, M. I., Bay, M., and Downie, J. S., "Evaluation of algorithms using games: The case of music tagging.." in *ISMIR*, pp. 387–392, 2009.

[35] Schreiber, H., "Improving genre annotations for the million song dataset.." in *ISMIR*, pp. 241–247, 2015.

[36] Bertin-Mahieux, T., Ellis, D., Whitman, B., and Lamere, P., “The million song dataset.” in *ISMIR*, 2011.

[37] Ioffe, S., and Szegedy, C., “Batch normalization: Accelerating deep network training by reducing internal covariate shift.” in *International Conference on Machine Learning (ICML)*, pp. 448–456, 2015.

[38] Dieleman, S., and Schrauwen, B., “Multiscale approaches to music audio feature learning.” in *International Society for Music Information Retrieval Conference (ISMIR)*, pp. 116–121, 2013.

[39] Van Den Oord, A., Dieleman, S., and Schrauwen, B., “Transfer learning by supervised pre-training for audio-based music classification.” in *Proc. of the International Society for Music Information Retrieval*, 2014.

[40] Liu, J.-Y., Jeng, S.-K., and Yang, Y.-H. (2016). “Applying topological persistence in convolutional neural network for music audio signals.” *arXiv preprint arXiv:1608.07373*.

[41] Güçlü, U., Thielen, J., Hanke, M., van Gerven, M., and van Gerven, M. A., “Brains on beats.” in *Advances in Neural Information Processing Systems*, pp. 2101–2109, 2016.

[42] Pons, J., Slizovskaia, O., Gong, R., Gómez, E., and Serra, X. (2017). “Timbre analysis of music audio signals with convolutional neural networks.” *EUSIPCO*.

[43] Jeong, I.-Y., and Lee, K., “Learning temporal features using a deep neural network and its application to music genre classification..” in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 434–440, 2016.

[44] Park, J., Lee, J., Park, J., Ha, J.-W., and Nam, J. (2017). “Representation learning of music using artist labels.” *arXiv preprint arXiv:1710.06648*.

[45] Choi, K., Fazekas, G., Sandler, M., and Kim, J., “Auralisation of deep convolutional neural networks: Listening to learned features.” in *ISMIR, Late Breaking / Demo*, pp. 26–30, 2015.

[46] Bittner, R. M., McFee, B., Salamon, J., Li, P., and Bello, J. P., “Deep salience representations for f0 estimation in polyphonic music.” in *ISMIR*, 2017.

[47] Choi, K., Fazekas, G., and Sandler, M. (2016). "Explaining deep convolutional neural networks on music classification." *arXiv preprint arXiv:1607.02444.*

[48] Erhan, D., Bengio, Y., Courville, A., and Vincent, P. (2009). "Visualizing higher-layer features of a deep network." *University of Montreal*, 1341, p. 3.

[49] Zeiler, M. D., and Fergus, R., "Visualizing and understanding convolutional networks." in *European conference on computer vision*, pp. 818–833, Springer, 2014.

[50] Nguyen, A., Yosinski, J., and Clune, J., "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images." in *IEEE CVPR*, pp. 427–436, 2015.

[51] Lee, H., Pham, P., Largman, Y., and Ng, A. Y., "Unsupervised feature learning for audio classification using convolutional deep belief networks." in *Advances in neural information processing systems (NIPS)*, pp. 1096–1104, 2009.

[52] Adavanne, S., and Virtanen, T., "Sound event detection using weakly labeled dataset with stacked convolutional and recurrent neural network." in *DCASE Workshop*, 2017.

[53] Adavanne, S., Drossos, K., Çakır, E., and Virtanen, T., "Stacked convolutional and recurrent neural networks for bird audio detection." in *EUSIPCO*, 2017.

[54] Malik, M., Adavanne, S., Drossos, K., Virtanen, T., Ticha, D., and Jarina, R., "Stacked convolutional and recurrent neural networks for music emotion recognition." in *Sound and Music Computing Conference (SMC)*, 2017.

[55] Lee, J., Park, J., Kum, S., Jeong, Y., and Nam, J., "Combining multi-scale features using sample-level deep convolutional neural networks for weakly supervised sound event detection." in *DCASE Workshop*, 2017.

[56] Kim, T., Lee, J., and Nam, J. (2017). "Sample-level cnn architectures for music auto-tagging using raw waveforms." *arXiv preprint arXiv:1710.10451.*

[57] He, K., Zhang, X., Ren, S., and Sun, J., "Deep residual learning for image recognition." in *CVPR*, 2016.

[58] Hu, J., Shen, L., and Sun, G. (2017). "Squeeze-and-excitation networks." *arXiv preprint arXiv:1709.01507.*

[59] Warden, P. (2017). "Speech commands: A public dataset for single-word speech recognition.." *Dataset available from* http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz.

[60] "TensorFlow Speech Recognition Challenge." https://www.kaggle.com/c/tensorflow-speech-recognition-challenge/leaderboard.

[61] Mesaros, A., Heittola, T., Diment, A., Elizalde, B., Shah, A., Vincent, E., Raj, B., and Virtanen, T., "Dcase 2017 challenge setup: tasks, datasets and baseline system." in *DCASE Workshop*, 2017.

[62] Gemmeke, J. F., Ellis, D. P., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., and Ritter, M., "Audio set: An ontology and human-labeled dataset for audio events." in *IEEE ICASSP*, 2017.

[63] Xu, Y., Kong, Q., Wang, W., and Plumbley, M. D. (2017). "Surrey-cvssp system for dcase2017 challenge task4." *DCASE Tech. Rep.*

[64] Bengio, Y., Courville, A., and Vincent, P. (2013). "Representation learning: A review and new perspectives." *IEEE transactions on pattern analysis and machine intelligence*, 35, pp. 1798–1828.

[65] Humphrey, E., Bello, J., and LeCun, Y. (2013). "Feature learning and deep architectures: new directions for music informatics." *Journal of Intelligent Information Systems*, 41, pp. 461–481.

[66] Henaff, M., Jarrett, K., Kavukcuoglu, K., and LeCun, Y., "Unsupervised learning of sparse features for scalable audio classification." in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pp. 681–686, 2011.

[67] Yeh, C.-C., Su, L., and Yang, Y.-H., "Dual-layer bag-of-frames model for music genre classification." in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 246–250, 2013.

[68] Vaizman, Y., McFee, B., and Lanckriet, G. (2014). "Codebook-based audio feature representation for music information retrieval." *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22, pp. 1483–1493.

[69] Wülfing, J., and Riedmiller, M., “Unsupervised learning of local features for music classification.” in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 139–144, 2012.

[70] Schlüter, J., and Osendorfer, C., “Music Similarity Estimation with the Mean-Covariance Restricted Boltzmann Machine.” in *Proc. of the International Conference on Machine Learning and Applications*, pp. 118–123, 2011.

[71] Hamel, P., and Eck, D., “Learning features from music audio with deep belief networks.” in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pp. 339–344, 2010.

[72] Schmidt, E. M., and Kim, Y. E., “Learning emotion-based acoustic features with deep belief networks.” in *Proc. of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 65–68, 2011.

[73] Choi, K., Fazekas, G., Sandler, M., and Cho, K., “Transfer learning for music classification and regression tasks.” ISMIR, 2017.

[74] Lee, J., and Nam, J. (2017). “Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging.” *IEEE Signal Processing Letters*, 24, pp. 1208–1212.

[75] Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R., “Signature verification using a “siamese” time delay neural network.” in *Advances in Neural Information Processing Systems (NIPS)*, pp. 737–744, 1994.

[76] Koch, G., Zemel, R., and Salakhutdinov, R., “Siamese neural networks for one-shot image recognition.” in *ICML Deep Learning Workshop*, vol. 2, 2015.

[77] Manocha, P., Badlani, R., Kumar, A., Shah, A., Elizalde, B., and Raj, B., “Content-based representations of audio using siamese neural networks.” in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[78] Lu, R., Wu, K., Duan, Z., and Zhang, C., “Deep ranking: Triplet matchnet for music metric learning.” in *ICASSP*, IEEE, 2017.

[79] Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., and Heck, L., “Learning deep structured semantic models for web search using clickthrough data.” in

Proc. of the 22nd ACM international conference on Conference on information & knowledge management, pp. 2333–2338, ACM, 2013.

[80] Frome, A., Corrado, G. S., Shlens, J., Bengio, S., Dean, J., Mikolov, T., et al., "Devise: A deep visual-semantic embedding model." in *Advances in neural information processing systems (NIPS)*, pp. 2121–2129, 2013.

[81] Kim, K. L., Kum, S., Park, C. L., Lee, J., Park, J., and Nam, J., "Building k-pop singing voice tag dataset: A progress report." in *Late Breaking Demo in the International Society for Music Information Retrieval Conf.*, 2017.

[82] Park, J., Lee, J., Park, J., Ha, J.-W., and Nam, J., "Representation learning of music using artist labels." in *ISMIR*, 2018.

[83] Logan, B., and Salomon, A., "A music similarity function based on signal analysis.." in *ICME*, 2001.

[84] Slaney, M., Weinberger, K., and White, W., "Learning a metric for music similarity." in *ISMIR*, 2008.

[85] Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., and Wu, Y., "Learning fine-grained image similarity with deep ranking." in *CVPR*, 2014.

[86] Hoffer, E., and Ailon, N., "Deep metric learning using triplet network." in *Int. Workshop on Similarity-Based Pattern Rec.*, Springer, 2015.

[87] Jansen, A., Plakal, M., Pandya, R., Ellis, D., Hershey, S., Liu, J., Moore, R., and Saurous, R., "Unsupervised learning of semantic audio representations." in *ICASSP*, IEEE, 2018.

[88] Lee, J., Park, J., and Nam, J., "Representation learning of music using artist, album, and track information." in *Machine Learning for Music Discovery Workshop, ICML*, 2019.

[89] Choi, J., Lee, J., Park, J., and Nam, J., "Zero-shot learning for audio-based music classification and tagging." in *ISMIR*, 2019.

[90] Böck, S., Krebs, F., and Widmer, G., "Accurate tempo estimation based on recurrent neural networks and resonating comb filters.." in *ISMIR*, 2015.

[91] Böck, S., Korzeniowski, F., Schlüter, J., Krebs, F., and Widmer, G., "Madmom: A new python audio and music signal processing library." in *Int. Conf. on Multimedia*, ACM, 2016.

[92] McFee, B., Raffel, C., Liang, D., Ellis, D., McVicar, M., Battenberg, E., and Nieto, O., "librosa: Audio and music signal analysis in python." in *14th Python in Science Conf.*, 2015.

[93] Kim, T., Lee, J., and Nam, J. (2019). "Comparison and analysis of samplecnn architectures for audio classification." *JSTSP*, 13.

[94] Hu, J., Shen, L., and Sun, G., "Squeeze-and-excitation networks." in *CVPR*, 2018.

[95] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A., "Going deeper with convolutions." in *CVPR*, 2015.

[96] Kingma, D. P., and Ba, J., "Adam: A method for stochastic optimization." in *ICLR*, 2015.

[97] Liang, D., Paisley, J., and Ellis, D., "Codebook-based scalable music tagging with poisson matrix factorization.." in *ISMIR*, 2014.

[98] Nam, J., Choi, K., Lee, J., Chou, S.-Y., and Yang, Y.-H. (2018). "Deep learning for audio-based music classification and tagging: Teaching computers to distinguish rock from bach." *Signal Processing Magazine*, 36.

[99] Zhai, A., and Wu, H.-Y. (2019). "Classification is a strong baseline for deep metric learning." *BMVC*.

[100] Reed, S., Sohn, K., Zhang, Y., and Lee, H., "Learning to disentangle factors of variation with manifold interaction." in *ICML*, 2014.

[101] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P., "Infogan: Interpretable representation learning by information maximizing generative adversarial nets." in *NeurIPS*, 2016.

[102] Lee, J., Bryan, N. J., Salamon, J., Jin, Z., and Nam, J., "Disentangled multidimensional metric learning for music similarity." in *ICASSP*, IEEE, 2020.

[103] Chopra, S., Hadsell, R., and LeCun, Y., "Learning a similarity metric discriminatively, with application to face verification." in *CVPR*, vol. 1, IEEE, 2005.

[104] Hadsell, R., Chopra, s., and LeCun, Y., "Dimensionality reduction by learning an invariant mapping." in *CVPR*, vol. 2, IEEE, 2006.

[105] Schultz, M., and Joachims, T., "Learning a distance metric from relative comparisons." in *NeurIPS*, 2004.

[106] Weinberger, K. Q., and Saul, L. K. (2009). "Distance metric learning for large margin nearest neighbor classification." *Journal of Machine Learning Research*, 10.

[107] Schroff, F., Kalenichenko, D., and Philbin, J., "Facenet: A unified embedding for face recognition and clustering." in *CVPR*, IEEE, 2015.

[108] Movshovitz-Attias, Y., Toshev, A., Leung, T. K., Ioffe, S., and Singh, S., "No fuss distance metric learning using proxies." in *ICCV*, IEEE, 2017.

[109] Wu, C.-Y., Manmatha, R., Smola, A. J., and Krahenbuhl, P., "Sampling matters in deep embedding learning." in *ICCV*, IEEE, 2017.

[110] Qian, Q., Shang, L., Sun, B., Hu, J., Li, H., and Jin, R., "Softtriple loss: Deep metric learning without triplet sampling." in *ICCV*, IEEE, 2019.

[111] Musgrave, K., Belongie, S., and Lim, S.-N. (2020). "A metric learning reality check." *arXiv preprint arXiv:2003.08505*.

[112] Ridgeway, K., and Mozer, M. C., "Learning deep disentangled embeddings with the f-statistic loss." in *NeurIPS*, 2018.

[113] Goldberger, J., Hinton, G. E., Roweis, S. T., and Salakhutdinov, R. R., "Neighbourhood components analysis." in *NeurIPS*, 2005.

[114] Wu, Z., Xiong, Y., Yu, S. X., and Lin, D., "Unsupervised feature learning via non-parametric instance discrimination." in *CVPR*, IEEE, 2018.

[115] Won, M., Chun, S., and Serra, X. (2019). "Toward interpretable music tagging with self-attention." *arXiv preprint arXiv:1906.04972*.

[116] Oh Song, H., Xiang, Y., Jegelka, S., and Savarese, S., "Deep metric learning via lifted structured feature embedding." in *CVPR*, IEEE, 2016.

[117] Maaten, L. V. D., and Hinton, G. (2008). “Visualizing data using t-SNE.” *Journal of machine learning research*, 9.

Acknowledgment

Thank you to everyone who has been with me during my journey of the thesis over the years. First of all, I thank my parents, aunt, and my sister's family. I will attend family events more often. And, I thank my advisor for showing all the required qualities I need as a scholar. I will continue to work as ever without forgetting your teaching. Thank you to everyone who worked together in the MAC lab. I spent the most time with them and received a lot of comfort. I will continue to be a person who can influence each other for good. Thank you to those who I worked with at Adobe, Naver, and NYU. I was able to learn different perspectives from different environments. If there is a chance, I will contribute what I have received. Thank you to all the people I met in the ISMIR based conference community. I was able to see an organization with a great culture and I will continue to contribute to the many benefits I have received as a student. Also, thank you to the doctoral dissertation committees who gave me detailed comments. I was able to see a warm heart for the junior researchers from them, and I will also become a warm hearted researcher for someone. Also, thank you to my friends in Seoul who stayed there without change until I finish my thesis program. I will also always be there. Lastly, special thanks to my parents, Juhan, Nick, the members of Neutune, and Saebyul. For a more enjoyable tomorrow.!

Acknowledgment in Korean

지난 수년간의 학위 과정 중 함께한 모든 분들에게 감사드립니다. 가장 먼저 부모님, 고모 그리고 누나네 가족 모두에게 큰 감사를 드리며 가족행사 더 잘 참석토록 하겠습니다. 그리고 학자로서 필요한 모든 자질을 몸소 보여주시는 지도교수님에게 감사드립니다. 향후 활동에서도 가르침 잊지 않고 정진토록 하겠습니다. 연구실에서 함께한 모든 분들에게도 감사드립니다. 힘든 때에 즐거울 때에 옆에서 가장 많은 시간을 함께 보내며 많은 위안을 받았습니다. 계속해서 서로 좋은 영향을 주고 받을 수 있는 사람이 되도록 하겠습니다. 인턴 등을 하며 짧지 않은 시간을 보낸 Adobe, Naver, NYU에서 함께한 분들에게도 감사드립니다. 다양한 환경에서 다양한 시각을 배울 수 있었습니다. 받은 혜택들 기회가 된다면 갚도록 하겠습니다. ISMIR를 기반으로 학회 커뮤니티에서 만난 인연들에게도 모두 감사드립니다. 훌륭한 문화를 가진 조직을 볼 수 있었고 스스로도 학생으로서 받은 여러 혜택들 계속 이어지도록 기여하도록 하겠습니다. 또한, 세심한 코멘트를 주시며 지도해주신 박사학위논문 심사위원 분들에게 감사드립니다. 후배 연구자를 위하는 마음을 볼 수 있었고, 저 또한 누군가에게 따듯한 연구자가 되도록 하겠습니다. 또한, 학위 과정을 마칠 때까지 그 자리에 그대로 있어준 서울 친구들에게도 감사드립니다. 저 또한 늘 그자리에 있도록 하겠습니다. 마지막으로, 부모님, 남주한 교수님, Nick, Neutune 멤버들, 그리고 새별누나에게 특별히 감사 인사 드립니다. 더 즐거운 내일을 위하여.!

Curriculum Vitae

Name : Jongpil Lee (이종필)

Date of Birth: June 10, 1988

E-mail : richter@kaist.ac.kr

Educations

2017. 9. – 2021. 2. Graduate School of Culture Technology, KAIST (Ph.D)

2015. 3. – 2017. 8. Graduate School of Culture Technology, KAIST (M.S.)

2007. 3. – 2015. 2. Department of Electronic Engineering, Hanyang University (B.S.)

2004. 3. – 2007. 2. Kyunggi High School

Career

2020. 1. – 2020. 3. Visiting Ph.D, Music and Audio Research Laboratory, New York University, New York, United States

2019. 6. – 2019. 9. Research Intern, Audio Research Group, Adobe Research, San Francisco, United States

2017. 7. – 2017. 9. Research Intern, Clova Artificial Intelligence Research, Naver, Seongnam, South Korea

2011. 2. – 2012. 11. KATUSA (Korean Augmentation To the United States Army) as a Mandatory Military Service, Camp Walker, Daegu, South Korea

Publications

1. **Jongpil Lee**, Nicholas J. Bryan, Justin Salamon, Zeyu Jin, and Juhan Nam, ”Metric Learning VS Classification for Disentangled Music Representation Learning”, *International Society of Music Information Retrieval Conference (ISMIR), 2020*

2. Seungheon Doh, **Jongpil Lee**, Tae Hong Park, and Juhan Nam, ”Musical Word Embedding: Bridging the Gap between Listening Contexts and Music”, *Machine Learning for Media Discovery Workshop, International Conference on Machine Learning (ICML), 2020*
3. Keunhyoung Luke Kim, **Jongpil Lee**, Sangeun Kum, Chae Lin Park, and Juhan Nam, ”Semantic Tagging of Singing Voices in Popular Music Recordings”, *IEEE/ACM Transactions on Audio, Speech and Language Processing, 2020*
4. **Jongpil Lee**, Nicholas J. Bryan, Justin Salamon, Zeyu Jin, and Juhan Nam, ”Disentangled Multidimensional Metric Learning for Music Similarity”, *International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2020*
5. Jeong Choi*, **Jongpil Lee***, Jiyoung Park, and Juhan Nam, ”Zero-shot Learning for Audio-based Music Classification and Tagging”, * Equally contributed *International Society for Music Information Retrieval Conference (ISMIR), 2019*
6. Saebyul Park, Taegyun Kwon, **Jongpil Lee**, Jeounghoon Kim, and Juhan Nam, ”A Cross-Scape Plot Representation for Visualizing Symbolic Melodic Similarity”, *International Society for Music Information Retrieval Conference (ISMIR), 2019*
7. Jeong Choi, **Jongpil Lee**, Jiyoung Park, and Juhan Nam, ”Zero-shot Learning and Knowledge Transfer in Music Classification and Tagging”, *International Conference on Machine Learning (ICML), Machine Learning for Music Discovery Workshop, 2019*
8. **Jongpil Lee**, Jiyoung Park, and Juhan Nam, ”Representation Learning of Music Using Artist, Album, and Track Information”, *International Conference on Machine Learning (ICML), Machine Learning for Music Discovery Workshop, 2019*
9. Taejun Kim, **Jongpil Lee** and Juhan Nam, ”Comparison and Analysis of SampleCNN Architectures for Audio Classification”, *IEEE Journal of Selected Topics in Signal Processing, 2019*
10. Juhan Nam, Keunwoo Choi, **Jongpil Lee**, Szu-Yu Chou and Yi-Hsuan Yang, ”Deep Learning for Audio-based Music Classification and Tagging”, *IEEE Signal Processing Magazine, 2019*
11. **Jongpil Lee**, Kyungyun Lee, Jiyoung Park, Jangyeon Park and Juhan Nam, ”Deep Content-User Embedding Model for Music Recommendation”,

https://arxiv.org/abs/1807.06786, 2018

12. Jiyoung Park*, **Jongpil Lee***, Jangyeon Park, Jung-Woo Ha and Juhan Nam, "Representation Learning of Music Using Artist Labels", * Equally contributed, *International Society for Music Information Retrieval Conference (ISMIR), 2018*

13. Jiyoung Park, Donghyun Kim, **Jongpil Lee**, Sangeun Kum and Juhan Nam, "A Hybrid of Deep Audio Feature and i-vector for Artist Recognition", *International Conference on Machine Learning (ICML), Joint Workshop on Machine Learning for Music, 2018*

14. Seungsoon Park, **Jongpil Lee** and Juhan Nam, "NEUROSCAPE: Artificial Soundscape Based on Multimodal Connections of Deep Neural Networks", *International Computer Music Conference (ICMC), 2018*

15. Taejun Kim, **Jongpil Lee** and Juhan Nam, "Sample-level CNN Architectures for Music Auto-tagging Using Raw Waveforms", *arXiv preprint arXiv:1710.10451, 2017*
IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018

16. **Jongpil Lee**, Jiyoung Park, Keunhyoung Luke Kim and Juhan Nam, "SampleCNN: End-to-End Deep Convolutional Neural Networks Using Very Small Filters for Music Classification"
Applied Sciences, 2018

17. **Jongpil Lee**, Taejun Kim, Jiyoung Park and Juhan Nam, "Raw Waveform-based Audio Classification Using Sample-level CNN Architectures", *Neural Information Processing Systems (NIPS), Machine Learning for Audio Signal Processing Workshop, 2017*

18. **Jongpil Lee**, Jiyoung Park, Sangeun Kum, Youngho Jeong and Juhan Nam, "Combining Multi-Scale Features Using Sample-level Deep Convolutional Neural Networks for Weakly Supervised Sound Event Detection", *Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop/Challenge, 2017*

19. **Jongpil Lee**, Jiyoung Park, Juhan Nam, Chanju Kim, Adrian Kim, Jangyeon Park and Jung-Woo Ha, "Cross-cultural Transfer Learning Using Sample-level Deep Convolutional Neural Networks", *Music Information Retrieval Evaluation eXchange (MIREX), 2017*
(1st place in the four K-POP tasks across all algorithms submitted so far)

20. Jiyoung Park, **Jongpil Lee**, Juhan Nam, Jangyeon Park and Jung-Woo Ha, ”Representation Learning Using Artist Labels for Audio Classification Tasks”,
Music Information Retrieval Evaluation eXchange (MIREX), 2017
(1st place in Music Mood Classification across all algorithms submitted so far)

21. KeunHyoung Luke Kim, Sangeun Kum, Chae Lin Park, **Jongpil Lee**, Jiyoung Park and Juhan Nam, ”Building K-POP Singing Voice Tag Dataset: A Progress Report”,
International Society for Music Information Retrieval Conference (ISMIR), Late Breaking/Demos, 2017

22. Dongwoo Suh, Kyungyun Lee, **Jongpil Lee**, Jiyoung Park and Juhan Nam, ”MUSIC GALAXY HITCHHIKER: 3D Web Music Navigation Through Audio Space Trained with Tag and Artist Labels”,
International Society for Music Information Retrieval Conference (ISMIR), Late Breaking/Demos, 2017

23. **Jongpil Lee** and Juhan Nam, “Multi-Level and Multi-Scale Feature Aggregation Using Sample-level Deep Convolutional Neural Networks for Music Classification”,
International Conference on Machine Learning (ICML), Machine Learning for Music Discovery Workshop, 2017

24. **Jongpil Lee**, Jiyoung Park, Keunhyoung Luke Kim and Juhan Nam, ”Sample-level Deep Convolutional Neural Networks for Music Auto-Tagging Using Raw Waveforms”,
Sound and Music Computing Conference (SMC), 2017

25. **Jongpil Lee** and Juhan Nam, “Multi-Level and Multi-Scale Feature Aggregation Using Pre-trained Convolutional Neural Networks for Music Auto-tagging”,
IEEE Signal Processing Letters, 2017

26. Hyeongseok Wi, Kyung hoon Hyun, **Jongpil Lee** and Wonjae Lee, “The Effect of DJs’ Social Network on Music Popularity”,
International Computer Music Conference (ICMC), 2016

27. **Jongpil Lee**, Taehyoung Kim, Sangeun Kum, Keunhyoung Luke Kim, Changheun Oh and Juhan Nam, “Investigation on Vocal Tags and Singer Similarity of K-pop”,
The Acoustical Society of Korea, 2016