

UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

Content-Based Music Recommendation with the LFM-1b Dataset and Sample-Level Deep Convolutional Neural Networks

Permalink

<https://escholarship.org/uc/item/3137p3v0>

Author

Platt, Devin

Publication Date

2017

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA, SAN DIEGO

**Content-Based Music Recommendation with the LFM-1b Dataset and
Sample-Level Deep Convolutional Neural Networks**

A thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Computer Science

by

Devin Platt

Committee in charge:

Professor Gert Lanckriet, Chair
Professor Sanjoy Dasgupta
Professor Lawrence Saul

2017

Copyright

Devin Platt, 2017

All rights reserved.

The thesis of Devin Platt is approved, and it is acceptable in quality and form for publication on microfilm:

Chair

University of California, San Diego

2017

TABLE OF CONTENTS

	Signature Page	iii
	Table of Contents	iv
	List of Figures	vi
	List of Tables	vii
	Acknowledgements	viii
	Abstract of the Thesis	ix
Chapter 1	Introduction	1
Chapter 2	Background	4
	2.1 Audio-based Music Classification	5
	2.1.1 Labels	5
	2.1.2 Features	6
	2.1.3 Model Architectures	9
	2.1.4 Transfer Learning	10
	2.2 Recommendation	11
	2.3 Audio-based Music Recommendation	14
Chapter 3	Dataset	16
Chapter 4	Architecture	19
Chapter 5	Latent Factor Models	21
	5.1 Methods	21
	5.2 Dataset for Implicit Feedback	23
	5.3 Number of tracks to train WMF model on	23
	5.4 Merging Tracks with Same Artist, Track Name	24
	5.5 Number of Latent Factors	25
	5.6 Number of Iterations	25
	5.7 Conclusions from Latent Factor Experiments	26
Chapter 6	Latent Factor Predictions	27
	6.1 Methods	27
	6.2 Results	30
	6.2.1 Quantitative Evaluation	30
	6.2.2 Qualitative Evaluation of Model	32
Chapter 7	Conclusions	35

Bibliography 37

LIST OF FIGURES

Figure 2.1:	Example of an audio waveform and corresponding log-amplitude mel-spectrogram.	7
Figure 2.2:	Diagram from Lee et al.	10
Figure 2.3:	An example user-item matrix, indicating explicit user ratings of retail products. (Image from Wikimedia ¹)	12
Figure 2.4:	Example of matrix factorization with 2 latent factors. (Image from Wikimedia ²)	13
Figure 3.1:	Dataset links for MSD and LFM-1b.	17

LIST OF TABLES

Table 3.1:	Comparison of LFM-1b and the Taste Profile Subset.	17
Table 5.1:	Dataset for Implicit Feedback.	23
Table 5.2:	Number of tracks to train WMF model on.	24
Table 5.3:	Merging Tracks with Same Artist, Track Name.	25
Table 5.4:	Number of Latent Factors.	25
Table 5.5:	Number of Iterations.	25
Table 6.1:	Recommendation Results	31
Table 6.2:	Tag Results	32
Table 6.3:	Track rankings for Nuthin’ But A “G” Thang by Dr. Dre, using ground truth vectors.	33
Table 6.4:	Track rankings for Nuthin’ But A “G” Thang by Dr. Dre, using predicted vectors.	33
Table 6.5:	Track rankings for Tootie For Cootie by Duke Ellington, using ground truth vectors.	33
Table 6.6:	Track rankings for Tootie For Cootie by Duke Ellington, using predicted vectors.	34
Table 6.7:	Track rankings for Polly by Nirvana, using ground truth vectors.	34
Table 6.8:	Track rankings for Polly by Nirvana, using predicted vectors.	34

ACKNOWLEDGEMENTS

I would like to thank Manoj Plakal for getting me started in computer audition by hosting me for an internship at Google Research. Additional thanks go to Ron Weiss, Dan Ellis, and the rest of the “Daredevil” team for providing guidance and teaching me much about techniques in the field.

I’d also like to thank Emanuele Coviello and Gert Lanckriet for mentorship during my time at Keevio and the subsequent year on the research team at Amazon Music.

Many thanks go to Keunwoo Choi for providing me with the audio the Million Song Dataset.

Thank you Piotr and Bartosz for the enlightening email discussions. Although I switched my focus for this paper from generative models to discriminative ones, I appreciate the correspondence. I’m sure you will have an excellent Masters thesis of your own soon, and I look forward to seeing your final work!

Finally, I would like to thank my committee: Lawrence Saul, Sanjoy Dasgupta, and Gert Lanckriet, for taking the time to review my thesis.

ABSTRACT OF THE THESIS

**Content-Based Music Recommendation with the LFM-1b Dataset and
Sample-Level Deep Convolutional Neural Networks**

by

Devin Platt

Master of Science in Computer Science

University of California San Diego, 2017

Professor Gert Lanckriet, Chair

Content-based music classification systems attempt to predict musical attributes of songs directly from their audio content. Commonly, the ground truth labels are explicitly identified traits such as the genre of a piece of music. Ground truth annotations can also be derived implicitly from user listening patterns, leading to content-based music recommendation systems.

We improve upon previous work in content-based music recommendation in two ways. One, we match the Million Song Dataset (MSD) to the recent LFM-1b dataset, which is

much larger than the standard Taste Profile Subset. Two, we train our model using deep convolutional architectures to predict latent factors directly from the audio of the music, instead of the standard practice of training on an intermediate time-frequency representation. We also evaluate the effectiveness of using latent factor prediction as a source task for tag prediction via transfer learning.

Chapter 1

Introduction

Recommendation forms an important component of today’s music streaming platforms. Collaborative filtering techniques are commonly used for recommending music, but suffer from the cold-start problem —if no one has listened to a song yet, the system won’t know how it should recommend that song. [Die16]

Previous work has shown that content-based methods, using the audio of each song, can provide sensible recommendations [vdODS13]. These methods are in fact incorporated into the algorithms for recommendation in at least one commercial streaming platform, Spotify, where they supplement the main collaborative filtering approach [Die14] [Joh15].

Public research in content-based music information retrieval faces some challenges due to copyright restrictions on musical recordings. This is compounded by the fact that machine learning approaches typically benefit from having very large datasets.

The Million Song Dataset [BMEWL11] (from now on referred to as MSD) offers a

solution to some of this challenge. It includes identifying information like the artist and track name for one million popular music songs, and additional metadata with partial coverage with the dataset. While the audio of this dataset cannot be made available online to the whole world (due to copyright), ~30-60 second audio excerpts can be downloaded from 7digital using API access they provide for researchers.¹

It has been argued that the MSD is actually still too small for some content-based music information retrieval tasks, and that the lack of a larger dataset is stifling further research [Die16]. However, content-based music recommendation and tagging tasks cannot even make use of the majority MSD, because there is only partial metadata coverage. Only half of tracks have tags from the LastFM Tag Subset and only about 250k tracks have any of the top 50 tags most frequently used in research experiments. Only about 380k tracks are included in the Taste Profile Subset of user listening events.

Recently the Department of Computational Perception at Johannes Kepler University has released LFM-1b, a massive dataset of more than one billion song listening events on LastFM [Sch16]. To the best of our knowledge, no work has yet been done using both this dataset and the MSD.

In this work, we match LFM-1b with 70 percent of the MSD, and demonstrate that information from this dataset provides better training signal than the Taste Profile Subset.

Our matching is published online ².

¹In my experience, the 7digital ids provided in the MSD no longer corresponded to the same tracks when querying 7digital. Thus, the only way to obtain the audio of this dataset is to contact a researcher who has done research with the dataset and ask them to share the data (this is much faster than 1 million API calls anyways).

²<https://github.com/devinplatt/lfm-1b-msd-matching>

We train our models using a variation of the recent architecture of Lee et al. [LPKN17], and publish the code used for our experiments online ³.

We also evaluate the effectiveness of using latent factor prediction as a source task for tag prediction via transfer learning.

³<https://github.com/devinplatt/ms-thesis>

Chapter 2

Background

Content based music recommendation falls at the intersection of recommendation and computer audition algorithms which try to extract information from audio. In the context of music, these tasks fall under the broader field of Music Information Retrieval (MIR).

There are many audio-based music information retrieval tasks. For example, the Music Information Retrieval Evaluation eXchange (MIREX) lists some following tasks

- Audio classification (into different genres or moods)
- Audio Beat Tracking
- Audio Chord Estimation
- Audio Tempo Estimation
- Drum Transcription
- Real-time Audio to Score Alignment (a.k.a Score Following)

- ... and many more ¹

In this work, we focus on tasks which attempt to predict global information about a track from its audio. Whereas tasks like transcription or chord estimation care about when a particular event happens in a song, in our tasks we only care about accurately predicting attributes that apply to the song as a whole.

In this section we discuss the problems of audio-based music classification and recommendation separately, then review work that deals with the merger of the two.

2.1 Audio-based Music Classification

2.1.1 Labels

Tagging a musical track presents a host of challenges. It is a multi-label classification task, so the typical methods of single label tasks must be adjusted. As noted in [Die16] the following challenges must be confronted:

- A large number of tags.
- Weak labels: often times, some positive labels are missing in the datasets. (This is especially critical at test time, since we cannot assume that a model is wrong to predict a tag the does not appear in the ground truth labels.)
- Label redundancy: some labels can be highly correlated.

¹see http://www.music-ir.org/mirex/wiki/MIREX_HOME

- Label sparsity: most labels apply to only a small subset of the data.

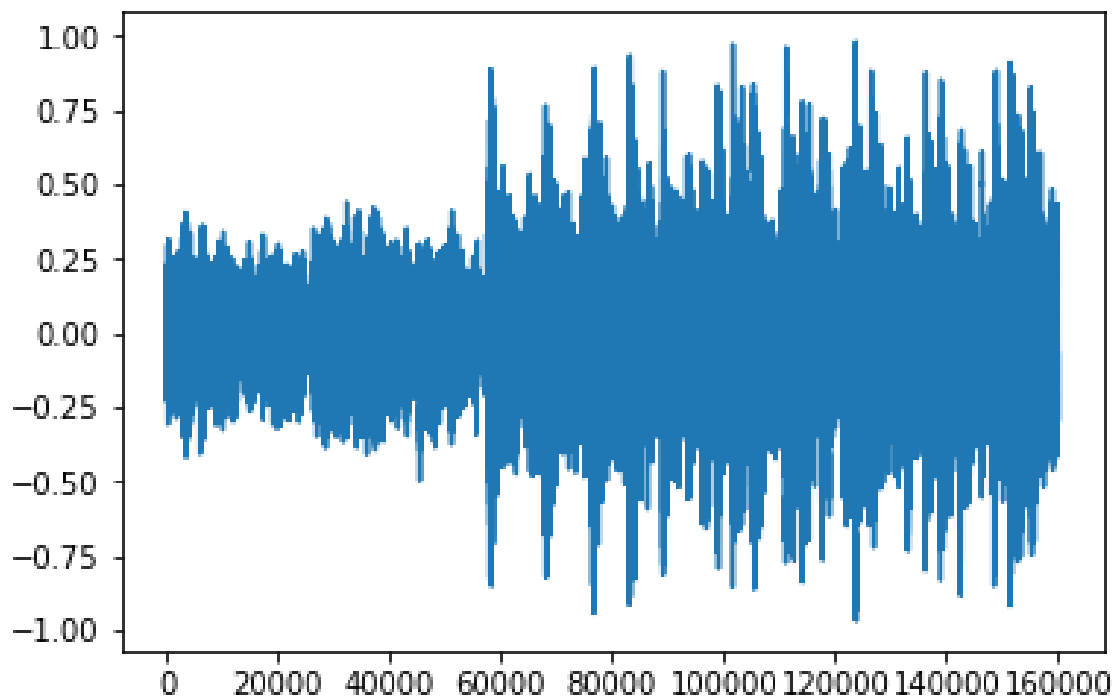
The best solution to these challenges is more data. The hope with more data is that we can overcome any noise caused by deficiencies in the labeling.

2.1.2 Features

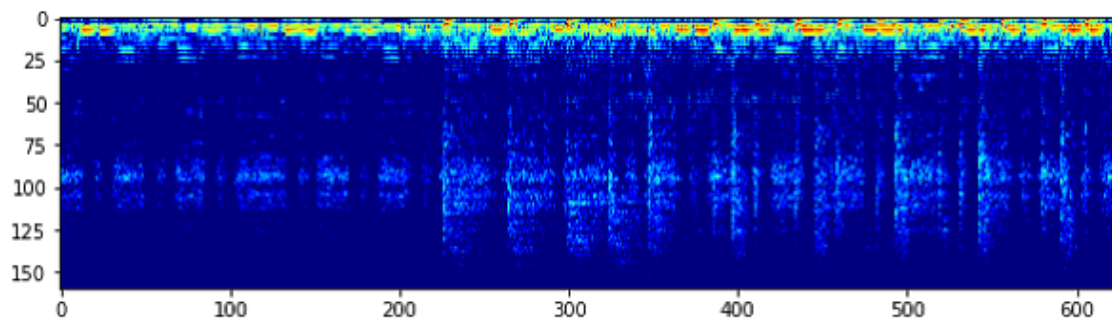
Audio, in its raw form, is stored on a computer as a series of integers representing a sound wave. Using Pulse Code Modulation (PCM), a continuous waveform is discretized in its time dimension according to its sample rate and is discretized in its amplitude according to its bit depth (See Figure 2.1). For example, a sequence of 16000 32-bit integers could represent one second of audio at a sample rate of 16kHz and a bit depth of 32.

Due to the Nyquist frequency theorem, a sampling rate of $2f$ is required to observe any frequency f in a signal. A human ear can hear frequencies between about 20Hz-20kHz. In CD quality audio, at sampling rate of 44.1kHz (twice the highest perceivable frequency) is used to cover the entire human perceivable frequency spectrum.

However, humans also perceive frequencies on a roughly logarithmic scale. Although there are roughly 10 octaves within 20Hz-20kHz, the first 9 of those exist in the 20Hz-10kHz range. Frequencies at the very upper end of human hearing tend not to be perceived as very pleasant or musical, and with aging the human hearing range is typically reduced, especially for frequencies in the upper range. Thus, when training computer models of audio, lower sample rates like 16000 or 22050 Hz are often used because they reduce dimensionality without discarding too much frequency information.



(a)



(b)

Figure 2.1: Example of an audio waveform and corresponding log-amplitude mel-spectrogram. The audio is sampled at a rate of 22050 Hz (this waveform has had its amplitude values stored as floating point values scaled to a range of $[-1, 1]$). This spectrogram uses 160 mel bins and half-overlapping windows of length 512 Hz.

Traditionally, mid-level features are extracted before sending the input to a classifier. Most commonly, audio is converted into a 2-dimensional time-frequency representation, such as a spectrogram (see Figure 2.1). Using the short-time Fourier transform (STFT), the intensities of different frequencies in short windows of audio can be computed. The values at these windows are concatenated together to form the spectrogram, showing changes in frequency intensities over time [Die16].

Typical window-lengths are about 20 ms, or 1/50th of a second. Frequencies are aggregated into mean-valued bins (the popular Librosa ² library defaults to 128 bins). These bins can follow a linear scale, but are often scaled according to the Mel scale, which accounts for the human logarithmic perception of pitch. The intensities of the spectrogram values may also be logarithmically scaled, to match human perception of loudness (known as the decibel scale).

Upsides of using mid-level features include:

- Your model can be shallower, and thus may train faster.
- You may be able to do more with less data, since your model only needs to learn higher level abstractions.

Downsides of using mid-level features include:

- Extra hyperparameter tuning. (I.e. how many mel bins do I use? What hop length do I use? etc.)

²<https://github.com/librosa/librosa>

- Code glue required to combine feature extraction and your model.
- Lower-level features are not learned directly in the service of the target task, so they might not perform quite as well.

With modern hardware and deep learning algorithms, it is possible to train models directly on raw audio. Recent work has shown that these models can perform comparably well or even better than models of similar complexity trained on mid-level features [DS14] [LPKN17].

2.1.3 Model Architectures

Convolutional neural networks are favored for reducing model complexity via shared weights and for providing temporal invariance. Audio can be processed with 1-dimensional convolutions, while for mel spectrogram input both 1-dimensional and 2-dimensional convolutions are possible.

In this paper we use the architecture of Lee et al. [LPKN17], who have obtained the best results to date on the typical benchmarks for music classification (Magnatagatune and the Million Song Dataset).

Lee et al. showed that it is beneficial to train fairly deep networks (up to 9 layers). Training deep convolutional models is made feasible by modern techniques. The *rectified linear unit (ReLU)* activation aids against the vanishing gradient problem common in deep networks. The *batch normalization* layer speeds up training by minimizing aspects of internal covariate shift, where layers struggle to handle changes in the output distributions

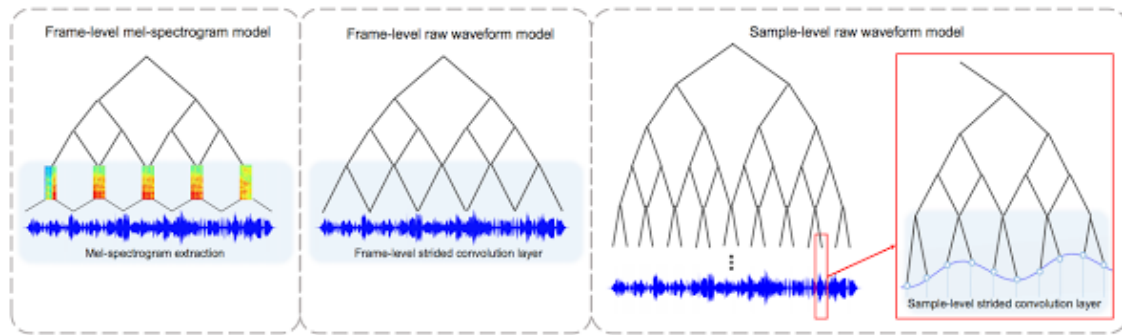


Figure 2.2: Diagram from Lee et al.

of their preceding layers.

The STFT spectrogram transformation is invariant to phase (time shift of periodic waveforms). To match the performance of spectrogram-based models, models of raw audio need to learn some method of phase invariance. Crucially, by operating at the sample-level (using an initial convolution of just a few audio samples), the architecture of Lee et al. becomes phase-invariant.

Due to the multi-label nature of the audio classification task, sigmoid activations are typically preferred to softmax at the output of a neural network.

2.1.4 Transfer Learning

Transfer learning provides an alternative to mid-level feature extraction. Practically speaking, it contains the same tradeoffs as using engineered features: we can train smaller models and with less data, but we need to have an extra feature extraction step. However, with transfer learning we can extract higher-level features than mid-level representations like the log-amplitude mel spectrogram, and there is the potential that models trained on

these features could perform better.

Features derived from models trained to predict tags from the LastFM subset of the Million Song Dataset were shown to be beneficial to tag prediction tasks on smaller datasets, such as Magnatagatune [vdODS14]. This work also showed that activations taken from intermediate layers of MLPs sometimes did better than those at the label-level as input to classifiers in a new task. Later work by Choi et al. shows that for deeper networks concatenating the activations from multiple layers can improve the performance [CFSC17].

2.2 Recommendation

Recommendation systems often rely on a *collaborative filtering* approach, where algorithms attempt to extract user preferences and item relations from user behavior. For example, users on an online retail website may provide ratings of products they have purchased. This *explicit feedback* can be used to drive future product recommendations for the users.

One popular collaborative filtering algorithm is *matrix factorization*. A matrix is formed with users as the rows and items as the columns. Cells contain a value reflecting a given user's history with a given item, like the rating they may have given a product (see Figure 2.3).

For industrial applications, these matrices are usually quite large, with possibly millions of rows and columns. Matrix factorization is a form of dimensionality reduction that

³https://commons.wikimedia.org/wiki/File:Collaborative_filtering.gif


























				
				
				
				
				
				

Figure 2.3: An example user-item matrix, indicating explicit user ratings of retail products. (Image from Wikimedia³)

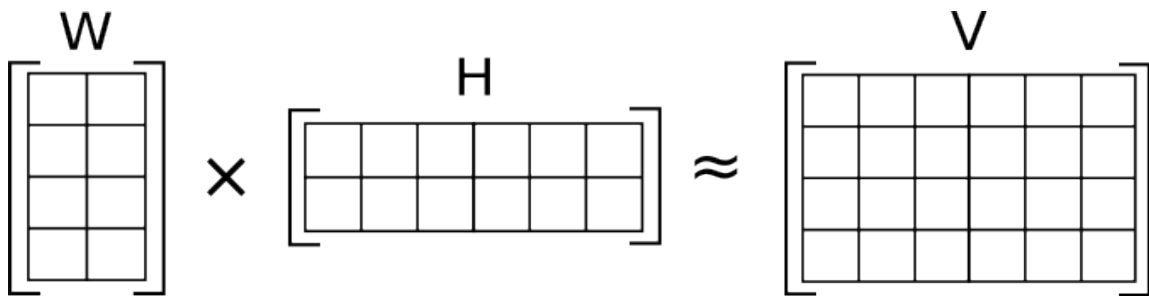


Figure 2.4: Example of matrix factorization with 2 latent factors. (Image from Wikimedia⁴)

attempts to approximate this large matrix as the product of two smaller matrices (See Figure 2.4).

The common number of inner dimensions of these two matrices is a hyperparameter of the algorithm. This is known as the number of *latent factors* of the model, because each dimension is expected to explain some broad trait of the data (eg. long historical fiction novels, men’s formal wear, rock music) that could describe a user preference or an item itself.

Rows for one of the matrices represent users, and columns for the other represent items. By computing the dot product of the latent factor vectors for any user-item pair, we create a score that can be used to rank user-item compatibility. (Alternative variations to the dot-product, such as the asymmetric cosine similarity, have also been proposed for the music recommendation task [Aio13].)

To calculate the matrix factorization approximation, stochastic gradient descent with the Alternating Least Squares method is often used because it is fast and easily parallelizable. Alternating Least Squares is an iterative method which modifies just one of the two

⁴<https://commons.wikimedia.org/wiki/File:NMF.png>

matrices at each iteration [KBV09].

The data available to build music recommendation systems are typically *implicit feedback*. In an online music streaming service, users provide their preferences implicitly according to their listening history. For the matrix factorization approach, the cells of the matrix will contain the number of times the given user has played the given song (sometimes this is simplified to a 1/0 representation of played/not played).

Specialized methods of matrix factorization exist for implicit feedback datasets. For explicit feedback datasets, unknown values are treated as missing, and optimization procedures make use of regularization to prevent overfitting. With implicit feedback datasets, unknown values can be dealt with by introducing a notion of confidence to predictions of user preferences [HKV08].

2.3 Audio-based Music Recommendation

It is widely accepted that collaborative filtering outperforms a purely content-based approach to recommendation [Sla11]. However, content-based methods can provide support in cases where collaborative filtering methods fail, such as when little listening data is available for a song.

Our work largely derives from van Den Oord et al. [vdODS13]. Using the Taste Profile Subset of the Million Song Dataset, they applied weighted matrix factorization to learn latent factors for users and songs, using the algorithm optimized for implicit feedback datasets of Hu et al. [HKV08]

The factors trained on the Taste Profile Subset had meaningful signal. A linear regression model van Den Oord et al. trained on the latent factors could predict LastFM song tags fairly well.

They then trained models to predict the latent factors from audio-based input (using mel spectral features), showing that CNN models on mel features worked much better than simpler models trained on an audio bag-of-words representation.

Our work builds on the work of van Den Oord et al. by using a substantially larger dataset, and training substantially larger models.

Chapter 3

Dataset

The main contribution of this thesis is the matching of two datasets: the Million Song Dataset [BMEWL11] and the LastFM-1b dataset [Sch16]. In this chapter we compare LFM-1b and the Taste Profile Subset, demonstrating the vast increase in data that LFM-1b provides. We also describe our method for matching MSD with LFM-1b in detail.

The Million Song Dataset (MSD) includes 30-60 second audio excerpts for one million tracks of popular music. It has previously been matched to music tags from LastFM and user listening events from an anonymous music streaming service, the Taste Profile Subset.

The LFM-1b dataset contains over one billion music listening events for over 120 thousand users of Last.fm. Each listening event includes an artist and track name, though the dataset is somewhat noisy in this respect: several instances of what is basically the same track may occur because of slight differences in the name.

Regardless, in terms of number of user interactions, this dataset is roughly 20 times

Table 3.1: Comparison of LFM-1b and the Taste Profile Subset.

Dataset	Number of Users	Number of Songs	Number of Interactions
LFM-1b	120 thousand	30 million (not unique)	1 billion
Taste Profile Subset	1 million	380 thousand (mostly unique)	48 million

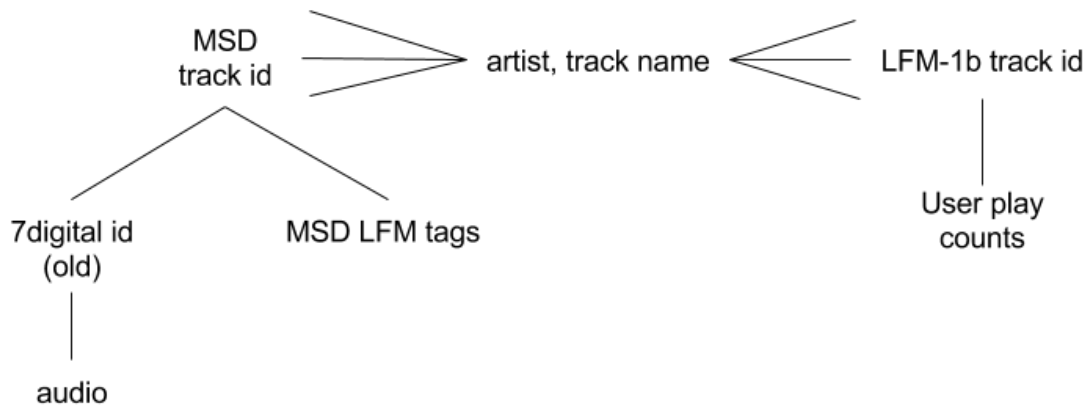


Figure 3.1: Dataset links. Tracks in the MSD and LFM-1b are matched by case-insensitive artist and track names. This matching is not perfectly one-to-one, but is close. LFM-1b tracks are tied to user listening data. MSD tracks are tied to audio (we were able to obtain audio samples for 99 percent of the tracks), and a subset are tied to tags from LastFM.

as large as the Taste Profile Subset (see Table 3.1), and can be matched to a much larger percentage of the tracks in the MSD.

To match LFM-1b to MSD, we used a simple matching strategy: case insensitive matching of the artist and track title (see Figure 3.1). No fuzzy matching algorithms like the Levenshtein distance were used. Using this naive approach, we matched tracks in LFM-1b to nearly 690k tracks in MSD, nearly twice as much as the 380k in the Taste Profile Subset. (Note that we match to about 630k songs, but 690k tracks because the Million Songs Dataset distinguishes between a song and a track. A song can be associated with multiple tracks.)

Even with this approach there are a few corner cases like artists with many “untitled” tracks, but this constitutes a very small fraction of the tracks.

It is possible to match more tracks, with more complicated approaches. Case-insensitive matching of artists with tracks in LFM-1b to tracks in MSD yields a 95 percent matching rate (For 95 percent of the tracks in MSD, there are at least some tracks in LFM-1b with the same artist). At the artist level, fuzzy string matching can be used. In this context, even if errors are made, they might not be too traumatic for collaborative filtering models, since tracks by the same artist are likely to be listened to by the same users.

Using case-insensitive matching of artist, with fuzzy title matching (set to conservative parameters by spot-checking), we matched about a third of this remaining 25 percent of tracks, or 8 percent of the entire dataset (on top of the 70 percent previously matched). However, we did not include these matched tracks for the rest of our experiments, because the improvement using the naive approach was already quite good and for our purposes the 8 percent additional tracks did not justify the additional complexity of the matching method. The rest of this paper uses just those matched using case-insensitive exact matching of artist and track.

Chapter 4

Architecture

In [vdODS13], the authors train a convolutional neural network on mel spectral features to predict latent factors (mel spectral features are a time-frequency representation of audio computed by running Fourier transforms on audio and using a logarithmic scaling of frequency and perhaps amplitude). Although the paper does not include many details, co-author Sander Dieleman mentions in a blog post that the paper used 2 convolutional layers followed by two fully connected layers [Die14].

In this paper, we use the best model from Lee et al. [LPKN17]. Known as a 3^9 model, it consists of 9 one-dimensional convolutional layers with length 3 kernels and length 3 max pooling. The input samples are reduced by a factor of 3 at each layer, down to a length of 1. An initial strided convolutional layer reduces the input to the appropriate size to allow for this reduction to exactly 1 sample in the last convolutional layer (in this case, we have a “sample-level” model: the input is $3^{10} = 59049$ raw audio samples, so the initial kernels

are also just 3 samples long).

Since their model is used for tagging, they append a dense output layer with an output for each tag and sigmoid activation, optimizing for binary cross entropy loss. In our case we replace this with linear activation and train using mean squared error loss (as used in [vdODS13]), with an output dimension for each latent factor. In our experiments we found it necessary to bound the output of our model, so we l2 normalized the output (matching the l2 normalization we applied to our target data).

Lee et al. trained on 59049 length segments of 29.1 second examples sampled at a 22050 Hz sample rate. We trained on 59049 length segments of 20 second examples at a 16000 Hz ¹ sample rate, mainly to save disk space.

Our work verifies the efficacy of the 3⁹ model, and shows that it can be trained to perform a regression task if the output is bounded with an l2 normalization layer.

¹Note that generally 16000 Hz is the lowest rate we would want to sample at because sampling at lower rates reduces the perceivable frequencies.

Chapter 5

Latent Factor Models

The Taste Profile Subset and LFM-1b both contain listening events: a tally of any time a user listened to a song. This kind of information is known as implicit feedback, because the users never explicitly indicated their musical preferences.

In this chapter we train latent factor models using these listening event data. We compare the quality of latent factors produced by LFM-1b versus the Taste Profile Subset, demonstrating quantitatively the superior quality of LFM-1b. We also analyze the effects of tuning different hyperparameters of our latent factor model. We summarize our results in a conclusion at the end of the chapter.

5.1 Methods

Van Den Oord et al. [vdODS13] use the modified weighted matrix factorization (WMF) technique of He et al. [HKV08], which is designed for implicit feedback datasets. In our

experiments we produce latent factors using this same algorithm with an implementation by one of the co-authors of Van Den Oord et al. ¹

We vary the inputs to our WMF model: our dataset (LFM-1b vs. Taste Profile Subset), the number tracks to train on, whether to merge similar tracks, the number of WMF factors, and the number of WMF iterations. Doing a full grid search would be extremely costly, so we vary one input at a time.

To compare the quality of latent factors, we set up a proxy task. We train a simple logistic regression classifier to predict LastFM tags ² in the MSD using the latent factors as input. As is standard in previous literature, we used the top 50 tags, which include labels such as ‘rock’, ‘pop’, ‘Hip-Hop’, ‘metal’, ‘electronic’, ‘female vocalists’, ‘dance’, and ‘80s’.

For the logistic regression experiments, we use the train, validation, test split released by Keunwoo Choi ³. Train, validation, and test sets are filtered to tracks found in the intersection of LFM-1b and the MSD LastFM tag subset. The validation and test sets are further filtered to tracks in the Taste Profile Subset. We trained each model for a maximum of 20 epochs, with an early stopping criterion of no improvement in validation loss between any two epochs.

¹<https://github.com/benanne/wmf>

²Note that although they come from the same source (music recommendation website LastFM), the LastFM tag subset of the MSD is a separate dataset from LFM-1b.

³https://github.com/keunwoochoi/MSD_split_for_tagging

Table 5.1: Dataset for Implicit Feedback. Hyperparameters: 40 latent factors, 40 iterations. AUC refers to the performance of a logistic regression classifier to predict musical tags from latent factors (using the AUC ROC metric).

Dataset	Tracks used for training WMF model	Number of tracks WMF model trained on	AUC
Taste Profile Subset	Taste Profile Subset	380k	.788
LFM-1b	LFM-1b / MSD matched tracks	660k	.8835

5.2 Dataset for Implicit Feedback

First, we verified that the latent factors produced from LFM-1b contain more supervisory signal than the Taste Profile subset (see Table 5.1). We obtained latent factors by training WMF models on the Taste Profile subset, and on LFM-1b (using user play counts every track that intersects with MSD).

We trained two separate logistic regression models on these factors, filtering training data to just tracks in Taste Profile Subset so that both logistic regression models use the same tracks (but not features) as training data.

Using LFM-1b, we see a 10-point improvement in AUC.

5.3 Number of tracks to train WMF model on

The LFM-1b dataset has information for 30 million tracks, too many to practically compute WMF on. However, most of the plays occur for the most popular tracks. We tried training a WMF model on the matched LFM-1b/MSD tracks plus all additional tracks in the top 2 million tracks (there was an overlap of $\sim 300k$ tracks).

Table 5.2: Number of tracks to train WMF model on. Hyperparameters: 40 latent factors, 40 iterations.

Dataset	Tracks used for training WMF model	Number of tracks WMF model trained on	AUC
LFM-1b	LFM-MSD matched tracks	660k	.8838
LFM-1b	LFM-MSD matched tracks + 2 million most popular tracks	2.3M	.8766

Surprisingly, it appears that adding these tracks degrades performance for this task (see Table 5.2). Though we had hoped that adding more tracks outside of the MSD would have produced a more robust model, it’s possible that using more tracks simply added too much noise to the data and causes it to underfit.

5.4 Merging Tracks with Same Artist, Track Name

A small percentage of tracks in LFM-1b actually have the same artist and track name when we normalize by case. It appears that merging these tracks (by summing playcounts and consolidating these equivalent tracks into one column) provides a small improvement in our proxy task (MSD tag prediction. See Table 5.3).

Note that this also solves the issue of linking a MSD track to a single latent factor.

Unfortunately, we tested this method late in our experimentation, so this method was not incorporated into our final latent factors used in later experiments for latent factor prediction from audio. In those experiments, we matched MSD tracks to an arbitrary latent factor of the set of possible matching LFM-1b tracks.

Table 5.3: Merging Tracks with Same Artist, Track Name. Hyperparameters: 40 latent factors, 40 iterations.

Dataset	Tracks used for training WMF model	Number of tracks WMF model trained on	AUC
LFM-1b	LFM-MSD matched tracks + 2 million most popular tracks	2.3M	.8766
LFM-1b	LFM-MSD matched tracks + 2 million most popular tracks, merging tracks with same artist and track name	2.3M - small amount	.8821

Table 5.4: Number of Latent Factors. Dataset: LFM-MSD matched tracks + 2 million most popular tracks. Hyperparameters: 40 iterations.

Number of Factors	AUC	Run Time
40	.8766	Not Recorded
50	.8812	170 minutes
80	.8918	267 minutes

5.5 Number of Latent Factors

We tested various values for the number of latent factors. We found that more factors generally improves performance, at the cost of longer runtime of the WMF algorithm. The WMF algorithm runtime scaled roughly linearly with the number of factors (see Table 5.4).

5.6 Number of Iterations

We also tested multiple values for the number of iterations of the WMF algorithm. Naturally, runtime scales linearly with the number of iterations (see table 5.5).

Table 5.5: Number of Iterations. Dataset: LFM-MSD matched tracks + 2 million most popular tracks. Hyperparameters: 80 factors.

Number of Iterations	AUC
40	.8918
80	.8934

5.7 Conclusions from Latent Factor Experiments

In this Chapter we demonstrated the superior supervisory signal provided by the data in LFM-1b versus the Taste Profile Subset. While part of the improvement could be due to the larger number of tracks available, we've demonstrated that there is a limit to how much more tracks really help the WMF model. We believe that much of benefit comes from the fact that LFM-1b has many more listening events than the Taste Profile Subset, and many more per user.

LFM-1b contains duplicate tracks with distinct identifiers, and we showed that merging instances of the same track led to modest performance gains. We also demonstrated modest gains in performance when increasing the number of factors or iterations, but overall showed that using at least 40 factors and 40 iterations is adequate.

In the next chapter we train models that predict latent factors from the audio of a track. In that chapter, we use our best performing latent factors, those trained on the largest subset of LFM-1b (MSD matched tracks plus the top 2 million tracks in LFM-1b), with 80 factors and 80 iterations. ⁴

Pre-computed factors similar to those in our experiments can be found on Github⁵.

⁴In hindsight, we probably should have trained a final model using the best parameters from each of our experiments: just the LFM-1b matched tracks, with 80 factors and 80 iterations.

⁵<https://github.com/devinplatt/ms-thesis>

Chapter 6

Latent Factor Predictions

In this chapter we describe our procedure for training a deep convolutional neural network to predict latent factors from audio. We then evaluate our model quantitatively in two ways. First, we evaluate it's effectiveness at the recommendation task. Second we evaluate the quality of its output as a signal for the tag prediction task. Finally, we observe track recommendations produced using the model's output to get better idea of what the model is learning.

6.1 Methods

We use the TensorFlow framework for our experiments. TensorFlow offers many useful features including implementations of state-of-the-art neural network algorithms, the TensorBoard visualization dashboard, and asynchronous multi-threaded queueing mechanisms for data I/O.

That last feature is especially important. In total, the MP3 files for the Million Song Dataset take several hundred gigabytes of disk space, far too large to fit in main memory of a regular workstation. Furthermore, small individual MP3 files are not ideal for fast reads.

We took the audio for matched tracks in MSD and our latent factors and compiled them into sharded TFRecord files. This amounted to about 700 shards with 1000 tracks each. To save disk space, we downsampled audio to 16000 Hz and stored 20 seconds of audio per track. Each shard took approximately 1.2 gigabytes.

We used separate shards for training, validation, and test data. We furthermore used separate shards for tracks also containing LFM tags, which we split into training, validation, and test sets following a split used in previous literature ¹. Our split can be found on Github².

We adapted our code from an open source implementation of the work of Lee et al. [LPKN17] for Magnatagatune³. Our implementation is available on GitHub². Our main change is the final layer of the network, which predicts latent factors instead of tags and optimizes for MSE instead of cross entropy loss. Since the network now performs a regression, we l2 normalized the output of the network to avoid issues of overactivation (NAN errors). We also modified the code to allow the enqueueing of multiple records at once asynchronously while the GPU trained the model. This decreased training time by an order of magnitude; training was not practically very achievable without efficient queuing.

¹https://github.com/keunwoochoi/MSD_split_for_tagging

²<https://github.com/devinplatt/ms-thesis>

³<https://github.com/tae-jun/sample-cnn>

When training our model we used random 59049 sample (3.7 seconds at 16kHz) crops from each 20 second audio sample. At evaluation time we took predictions for the first 5 adjacent 59049 sample windows (18.5 seconds total) and averaged them.

Our optimization procedure follows the similar methods of Lee et al.: optimize with SGD and momentum. Their methods began with an initial learning rate of 0.01, then reduced the learning rate by a factor of 5 when validation loss no longer improves, doing this a total of 4 times down to a final learning rate of 0.000016. They used a batch size of 50.

We used a smaller batch size of 32, because our 8GB GPU could not handle larger batch sizes. We started with a higher learning rate (0.5) because it sped up training. Although we ran for multiple stages like Lee et al., we saw only a small decrease in validation error, at the cost of training for twice as long. In our evaluation, we choose the best model from the initial learning rate stage.

Training for the initial learning rate stage took 3 days on an Nvidia 1070 GTX GPU. Each epoch took approximately 2 hours.

6.2 Results

6.2.1 Quantitative Evaluation

Recommendation

We evaluate our model’s prediction ability by using the predictions for music recommendation, following the methods of [vdODS13]. For each of the 120k users in LFM-1b, we rank a test set of 74k tracks. Following the Million Song Dataset Kaggle challenge⁴, we consider a recommendation “correct” if the user listened to a track. Our objective is to rank all tracks the user listened to as high as possible.

We report (Table 6.1) both AUC (AUCROC) and mean average precision cut off at 500 recommendations (mAP@500), which was the metric used in the Kaggle challenge. AUC and AP@500 are averaged across the 74k users.

Unlike the Kaggle challenge, we trained our latent factor models using all users and tracks (including test tracks), since the proximate task here is latent factor prediction, not recommendation. Thus, recommendations from the ground truth latent factors form an upper bound on what we expect to achieve from predicting the latent factors.

For each user, we create a user factor vector by multiplying their row of the listening events matrix with a matrix of predicted song factors, and then normalizing the resulting vector. We do this using only listening events and predicted song factors for songs in the training set. The user factor vector is effectively a weighted average of the factors of the

⁴<https://www.kaggle.com/c/msdchallenge>

Table 6.1: Recommendation Results

Model	mAP @ 500	AUC
Random	.000004	.5
Sample-level Raw Audio	0.001	.62
Upper Bound	.04	.96

training songs that user listened to.

We then multiply the user factor vector with each test song factor vector (also normalized) to get that song’s score for that user. These scores (between 0 and 1, with 1 indicating a top recommendation) are used to rank track recommendations for the user. From these scores we calculated per-user AUC using the scikit-learn Python package. We calculated each user’s average precision using the formula:

$$ap@n = \sum_{k=1}^{500} P(k) / \min(m, 500)$$

where m is the number of test tracks that the user listened to, and $P(k)$ is the precision at k : the number of tracks from positions 1 to k in the ranking that the user listened to.

Tag prediction

We previously showed a performance of .89 AUC in predicting MSD tags from latent factors. This is very close to the .90 AUC reached in Lee et al. [LPKN17] predicting tags from audio using a deep CNN.

Since the number of tracks with user listening events is so much greater than the number of tracks with tags, we wondered if features from the latent factor prediction model could perform comparatively well at the tag prediction task. Can latent factors be used as a source task for transfer learning?

Table 6.2: Tag Results

Model	AUC
CNN trained to predict tags from audio	0.9055 (as reported in Lee et al.)
Logistic regression on latent factors	0.8934
Logistic regression on predicted latent factors	0.8477

The results (Table 6.2) show that although predicted latent factors do not outcompete training to predict tags directly from audio (even with twice as much data), they do perform quite well.

6.2.2 Qualitative Evaluation of Model

To gain a better understanding of what the model is learning, we observe track recommendations produced using its output, and compare it to recommendations produced using the ground truth latent factor vectors (tables are included at the end of this section). We ranked tracks in the training set because it offered a larger pool of tracks to pick from. Although this does not test the model’s ability to generalize, it still gives us a peek at how it is working.

We can see that the model identifies some acoustic similarities. Sometimes it is good at recommending music in the same genre (such as with rap). Other times though, it ventures wildly out of genre, like when it recommends a song by The Riveiras for a Duke Ellington track (possibly because both employ brass instruments and drums).

The similarity scores are much closer for the predicted vectors than the ground truth vectors. Of course, the model is directly optimized to predict latent factor values, not song-similarities.

Table 6.3: Track rankings for Nuthin’ But A “G” Thang by Dr. Dre, using ground truth vectors.

Score	Track	Artist
1.000	Nuthin’ But A “G” Thang	Dr. Dre
0.925	Xxplosive	Dr. Dre
0.920	The Watcher	Dr. Dre
0.896	So Many Tears	2Pac
0.890	Big Poppa	The Notorious B.I.G.
0.887	Bang Bang	Dr. Dre
0.867	Let’s Get High	Dr. Dre
0.867	Eazy-Duz-It	Eazy-E
0.866	Protect Ya Neck	Wu-Tang Clan
0.863	Old School	2Pac

Table 6.4: Track rankings for Nuthin’ But A “G” Thang by Dr. Dre, using predicted vectors.

Score	Track	Artist
1.000	Nuthin’ But A “G” Thang	Dr. Dre
0.999	Break North	Ultramagnetic Mcs
0.999	Bounce	Diamond D
0.999	Underground G’s	5th Ward Boyz
0.999	What You Want	MC Breed
0.999	M.P.E.	Public Enemy
0.998	T.R.’z (tunnel Vision Album Version)	Tunnel Rats
0.998	Form Of Intellect	Gang Starr
0.998	Hustlers And Killers	Nas
0.998	Gold	Nice & Smooth

Table 6.5: Track rankings for Tootie For Cootie by Duke Ellington, using ground truth vectors.

Score	Track	Artist
1.000	Tootie For Cootie	Duke Ellington
0.756	Street Of Dreams	Sarah Vaughan
0.720	Just One More Chance	Dinah Washington
0.716	That’s Love	Art Pepper
0.714	St. Louis Blues	Ella Fitzgerald
0.708	Call The Police	Nat King Cole
0.704	Smoke Gets In Your Eyes	Nat King Cole
0.703	Harbor Lights	Dinah Washington
0.703	Red Bank Boogie	Count Basie
0.701	Cherokee	Dee Dee Bridgewater

Table 6.6: Track rankings for Tootie For Cootie by Duke Ellington, using predicted vectors.

Score	Track	Artist
1.000	Tootie For Cootie	Duke Ellington
0.999	's Wonderful	Ray Conniff
0.998	True Love Is Hard To Find	The Rivas
0.998	Goggles	Dick Clark
0.998	Love Is Just Around The Corner	Paul Anka
0.998	I Told You So	Jimmy Jones
0.998	Rhythm 'n' Blues	The McGuire Sisters
0.998	Hey You	Conway Twitty
0.998	Can I Go Home	Bo Diddley
0.998	Easy To Remember	The Rivas

Table 6.7: Track rankings for Polly by Nirvana, using ground truth vectors.

Score	Track	Artist
1.000	Polly	Nirvana
0.993	Territorial Pissings	Nirvana
0.933	Killing In The Name	Rage Against The Machine
0.923	Today	The Smashing Pumpkins
0.895	Guerrilla Radio	Rage Against The Machine
0.885	Iron Man	Black Sabbath
0.873	Sleep Now In The Fire	Rage Against The Machine
0.871	With Or Without You	U2
0.868	Dumb	Nirvana
0.846	Wouldn't It Be Nice	The Beach Boys

Table 6.8: Track rankings for Polly by Nirvana, using predicted vectors.

Score	Track	Artist
1.000	Polly	Nirvana
0.990	If That's Alright	Uncle Tupelo
0.989	Adiabatic	Prolapse
0.988	Take It Or Leave It	The Strokes
0.987	You're Not Alone	Home Grown
0.987	Undercovers On	Rival Schools
0.987	Baby I'm Just A Fool	Spiritualized
0.986	Monday	Fairmont
0.986	Alegato Meridional	Grupo De Expertos Solynieve
0.986	Modern Way	Kaiser Chiefs

Chapter 7

Conclusions

In this work we merged the LFM-1b dataset with the Million Song Dataset, achieving 70 percent coverage. We showed that the LFM-1b dataset provides substantially better supervisory signal than the Taste Profile Subset.

Using this dataset, a shallow model trained on ground truth latent factors was comparably effective at predicting tags to a deep model trained to predict tags from audio. The implications for this result are that in commercial music tagging systems with access to high quality user listening data, shallow factor-to-tag models could replace or supplement audio-to-tag models when user listening data is available for a track. This result was not as apparent with the Taste Profile Subset data.

We also demonstrated that the sample-level architecture of Lee et al. is effective for the latent factor prediction task. We found it necessary to bound our output with an l2 normalization layer. Dieleman indicates that he was successful both using and not using l2

normalization on the output when training shallower mel spectrum-based models [Die14]. We speculate that bounding the output may be necessary when training regressions on very deep models with linear and ReLu-type activations.

We showed that when the number and quality of user listening events is high, latent factor prediction can be used as source task for music tag prediction via transfer learning. Although still not as performant as a model trained to predict tags directly from audio, such a model could be effective when labels are in short supply, or could be used as supplementary input to an ensemble model.

Future work could include experimenting with the amount of available ground truth tags to see when latent factor prediction become beneficial as a source task for tag prediction. We would also like to try ensemble models trained to predict tags using both audio and latent factor input.

Bibliography

- [Aio13] Fabio Aioli. Efficient top-n recommendation for very large scale binary rated datasets. In *ACM conference on Recommender systems (RecSys)*, 2013.
- [BMEWL11] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *International Society for Music Information Retrieval (ISMIR)*, 2011.
- [CFSC17] Keunwoo Choi, Gyorgy Fazekas, Mark Sandler, and Kyunghyun Cho. Transfer learning for music classification and regression tasks. In *International Society of Music Information Retrieval (ISMIR)*, 2017.
- [Die14] Sander Dieleman. Recommending music on spotify with deep learning, 2014. (Blog Post). URL: <http://benanne.github.io/2014/08/05/spotify-cnns.html>.
- [Die16] Sander Dieleman. *Learning feature hierarchies for musical audio signals*. PhD thesis, 2016.
- [DS14] Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014.
- [HKV08] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *IEEE International Conference on Data Mining*, 2008.
- [Joh15] Chris Johnson. From idea to execution: Spotify’s discover weekly, 2015. (Slides). URL: <https://www.slideshare.net/MrChrisJohnson/from-idea-to-execution-spotifys-discover-weekly>.
- [KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 8:42, 2009.

- [LPKN17] Jongpil Lee, Jiyoung Park, Keunhyoung Luke Kim, and Juhan Nam. Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms. In *Sound and Music Computing Conference (SMC)*, 2017.
- [Sch16] M. Schedl. The lfm-1b dataset for music retrieval and recommendation. In *Proceedings of the ACM International Conference on Multimedia Retrieval (ICMR 2016)*, 2016.
- [Sla11] M. Slaney. Web-scale multimedia analysis: Does content matter? *IEEE MultiMedia*, 18(2):12–15, 2011.
- [vdODS13] Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Neural Information Processing Systems (NIPS)*, 2013.
- [vdODS14] Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Transfer learning by supervised pre-training for audio-based music classification. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2014.