Automatische hiërarchische kenmerkextractie voor muzikale geluidssignalen

Learning Feature Hierarchies for Musical Audio Signals

Sander Dieleman

Promotoren: prof. dr. ir. J. Dambre, dr. ir. B. Schrauwen
Proefschrift ingediend tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen: Computerwetenschappen

Vakgroep Elektronica en Informatiesystemen
Voorzitter: prof. dr. ir. R. Van de Walle
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2015 - 2016

UNIVERSITEIT GENT

ISBN 978-90-8578-874-4
NUR 984
Wettelijk depot: D/2016/10.500/6

Dankwoord

Ik wil graag mijn promotoren, eerst Benjamin Schrauwen en later ook Joni Dambre, bedanken voor de kans en de financiële ondersteuning om dit onderzoek te kunnen doen, maar vooral ook voor de vrijheid die ze mij gegeven hebben om hierbij mijn eigen traject uit te tekenen. Na mijn masterproef bij het Reservoir Lab over nootherkenning voor elektrische gitaren kreeg ik de kans om me voltijds toe te leggen op de computationele analyse van muzikale geluidssignalen. Uiteindelijk was het niet evident om te mogen werken rond een dergelijk onderwerp, waarover in de onderzoeksgroep slechts beperkte expertise aanwezig was en waarvan de impact en commerciële relevantie toentertijd nog zwaar onderschat werden. Desondanks mocht ik toch gewoon mijn *goesting* doen, zolang het maar doordacht en vooral ambitieus was.

Ik wil ook mijn collega's bedanken: Aäron, Antonio, David, Eric, Francis, Ira, Jeroen, Jonas, Juan-Pablo, Ken, Lionel, Michiel, Philemon, Pieter, Pieter-Jan, Thibault en Tim, en ook Brahim, Elias, Karel en Tom. Het Reservoir Lab is altijd een hechte groep geweest, getuige de vele activiteiten die we na de uren samen met onze collega's van HES organiseerden.

David, die daarvoor ook al mijn masterproef mee begeleid had, heeft mij in het begin van mijn doctoraat zeer veel meegegeven, en ook zijn technische ondersteuning heb ik zeer gewaardeerd. Pieter-Jan, altijd goedgezind, stond steeds klaar als er iets moest nagelezen worden, als er spelletjes gespeeld werden na de uren en ook gewoon, als iemand hem nodig had. Iets wat hij nu ook weer bewees, toen ik hem de dag voor de deadline vroeg of hij deel wou uitmaken van mijn examencommissie. Bedankt daarvoor, en ook voor de vier jaar die we samen doorbrachten in ons kantoor.

Heel veel dank gaat uit naar Aäron, met wie ik veel ideeën heb kunnen uitwisselen, en ook voor onze samenwerking op verschillende projecten. Dankzij die samenwerking werd de interesse van de industrie in ons werk

gewekt, en hebben we beiden een mooie stage kunnen doen. Van Philemon heb ik heel veel bijgeleerd over deep learning. Bedankt voor de vele discussies over RBMs en DBNs, en voor de geduldige, doordachte antwoorden op mijn soms ondoordachte vragen. Francis, bedankt voor het organiseren van de robotcompetitie, voor de SAP-hulp, en voor ijs en brownies. *Ira, thanks for satisfying my sweet tooth with exorbitant amounts of chocolate.*

Verder wil ik Aäron, Ira, Jeroen, Jonas, Lionel en Pieter ook nog eens bedanken voor de vlotte samenwerking bij onze deelname aan de National Data Science Bowl, en Joni om ons samen aan dit probleem te laten werken. Ik bedank ook Marnix, Ronny B., Michiel R. en Jeroen O. voor hun technische en logistieke ondersteuning.

Thanks to Erik, Chris, Ahmad, Rohan, Nikhil, Vidhya, Ed, Esh, Charlie and the rest of the Spotify discovery team for a great internship and an awesome three months in New York. Thanks also to my new colleagues at DeepMind, for their support and company during the many late nights filled with thesis writing.

Graag bedank ik ook mijn vrienden, in het bijzonder Jeffrey, Ioanna, Joni DM en de voltallige VGMS voor hun steun. *My thanks also go out to my colleagues and friends from the got-djent.com team, and to my collaborators on the Lasagne project.*

Aan het einde van mijn educatieve loopbaan bedank ik ook graag iedereen die daar in meer of mindere mate toe bijgedragen heeft; in het bijzonder Juf Claire, die mij vanaf het prille begin zelf liet onderzoeken hoe de wereld in elkaar zit.

Nogmaals bedankt aan Gert De Cooman, Tijl De Bie, Bart Dhoedt, Mike Kestemont en Pieter-Jan Kindermans om deel te willen uitmaken van mijn examencommissie.

Tot slot bedank ik mijn mama, mijn papa, mijn zus Joke, mijn grootouders en de rest van mijn familie, voor het warme nest waarin ik heb mogen opgroeien en voor hun onvoorwaardelijke en onuitputtelijke steun in alles wat ik doe.

Sander Dieleman

Examencommissie

prof. Joni Dambre, promotor
Vakgroep ELIS, Faculteit Ingenieurswetenschappen en Architectuur
Universiteit Gent

dr. Benjamin Schrauwen, co-promotor
Autodesk
San Francisco, VS

prof. Gert De Cooman, voorzitter
Vakgroep ELIS, Faculteit Ingenieurswetenschappen en Architectuur
Universiteit Gent

prof. Tijl De Bie
Vakgroep ELIS, Faculteit Ingenieurswetenschappen en Architectuur
Universiteit Gent

prof. Bart Dhoedt, secretaris
Vakgroep INTEC, Faculteit Ingenieurswetenschappen en Architectuur
Universiteit Gent

dr. Mike Kestemont
Centrum voor Computerlinguïstiek en Psycholinguïstiek
Departement Letterkunde
Universiteit Antwerpen

dr. Pieter-Jan Kindermans
Machine Learning Group
Technische Universität Berlin

Samenvatting

Muziek wordt steeds vaker in digitale vorm geconsumeerd. Het is voor velen de gewoonte om muziek te kopen in de vorm van digitale geluidsbestanden in plaats van een fysiek medium, of om muziek te *streamen* via het internet. De digitale distributie en consumptie van muziek heeft veel mogelijkheden voor automatisering gecreëerd. Het doorzoeken van grote muziekcatalogi kan bijvoorbeeld gestroomlijnd worden door het analyseren van de bijbehorende metadata en de inhoud van de geluidssignalen, om de muziek op verschillende manieren te classificeren en organiseren. Dit is zowel voor de luisteraar als voor de verkoper voordelig: de luisteraar wordt geholpen bij het beheren van zijn persoonlijke collectie, en de verkoper kan zijn inkomsten verhogen door het voor potentiële klanten eenvoudiger te maken om te vinden waar ze naar op zoek zijn.

Music information retrieval

Het onderzoeksgebied dat *music information retrieval (MIR)* genoemd wordt, handelt over de analyse van muziek en bijbehorende metadata met behulp van computationele methoden, en combineert ideeën uit de musicologie, signaalverwerking en machinaal leren. Onderzoekers die werken rond *inhoudsgebaseerde* MIR zijn in het bijzonder geïnteresseerd in het extraheren van muzikale informatie uit geluidssignalen. Dit is een uitdagende taak: er is een vrij grote semantische kloof tussen de golfvormen van het geluidssignaal die een muzieknummer voorstellen enerzijds, en de hoogniveau-aspecten van de muziek die onze voorkeur beïnvloeden anderzijds (zoals bv. genre, instrumentatie, ...).

Deze thesis

Muzikale geluidssignalen zijn inherent hiërarchisch gestructureerd, en dit op verschillende manieren. In het onderzoek dat in deze thesis beschreven wordt, heb ik methoden voor inhoudsgebaseerde MIR ontwikkeld die deze hiërarchische structuur uitbuiten. Hiervoor heb ik hoofdzakelijk gebruik gemaakt van technieken uit *deep learning*, een groeiend onderzoeksveld binnen het machinaal leren. Het idee van deep learning is om modellen te bouwen die data voorstellen op verschillende abstractieniveaus, en die in staat zijn om deze voorstellingen op een autonome manier uit de data zelf te extraheren. Deep learning modellen bestaan doorgaans uit verscheidene verwerkingslagen die een hiërarchie vormen: elke opeenvolgende laag extraheert een steeds abstractere voorstelling van de invoer, en bouwt hierbij verder op de voorstelling die geëxtraheerd werd door de vorige laag. Deze modelstructuur sluit nauw aan bij de hiërarchische structuur van muziek.

Muziekclassificatie met een *pre-trained* convolutioneel neuraal netwerk

De *Million Song Dataset* bevat audiogebaseerde kenmerken en metadata voor één miljoen nummers. We hebben een convolutioneel neuraal netwerk gebouwd dat getraind werd op deze dataset om artiestherkenning, genreherkenning en toonaardherkenning te doen. Het netwerk is ontworpen om de geluidskenmerken samen te vatten over muzikaal relevante tijdschalen. Omdat er weinig beschikbare labels zijn is het niet haalbaar om het netwerk op de volledige dataset te trainen op een gesuperviseerde manier. We steunen daarom op *unsupervised pre-training* om alle data nuttig te kunnen gebruiken. Daarna gebruiken we de parameters die op deze manier geleerd werden om een convolutioneel neuraal netwerk met dezelfde architectuur te initialiseren. Dit netwerk wordt dan verder afgesteld op een gelabelde subset van de data voor elke afzonderlijke taak.

Automatisch leren van audiogebaseerde kenmerken voor muziek op meerdere tijdschalen

Inhoudsgebaseerde MIR-taken worden meestal opgelost in twee stappen: eerst worden kenmerken geëxtraheerd uit muzikale geluidssignalen, en die worden dan gebruikt als invoer voor een regressie- of classificatiemodel. Deze kenmerken kunnen manueel ontworpen zijn, of geleerd op basis van data. Deze eerste aanpak was tot voor kort dominant, maar tegenwoordig krijgt het automatisch leren van zulke kenmerken steeds meer aandacht binnen

de MIR-gemeenschap. Er is ook steeds meer interesse in voorstellingen van muzikale geluidssignalen op meerdere tijdschalen, die in verschillende mate relevant zijn voor diverse MIR-taken.

Recent werden een aantal positieve resultaten geboekt op gebied van het automatisch leren van kenmerken met behulp van zeer eenvoudige en snelle algoritmen, zoals K-means. Hierdoor geïnspireerd, stellen we drie verschillende architecturen voor voor het leren van audiokenmerken op meerdere tijdsschalen op basis van het *spherical K-means* algoritme. We evalueren deze architecturen voor het automatisch *taggen* van muziek, het leren van een gelijkaardigheidsmetriek (*similarity metric*) en voor genreherkenning. Deze evaluatie gebeurt aan de hand van de Magnatagatune en GTZAN datasets, alsook bepaalde subsets van de Million Song Dataset.

Diepe inhoudsgebaseerde muziekaanbeveling

Automatische muziekaanbeveling wordt een steeds relevanter probleem, aangezien zeer veel muziek tegenwoordig onder digitale vorm verkocht en beluisterd wordt. De meeste aanbevelingssystemen werken op basis van *collaborative filtering*. Deze aanpak lijdt echter onder het zogenaamde *cold start*-probleem: hij faalt wanneer geen gebruiksdata beschikbaar is, en is dus niet geschikt voor het aanbevelen van nieuwe en minder populaire nummers.

We stellen voor om een *latent factor model* te gebruiken voor aanbeveling, en om de latente factoren te voorspellen uit muzikale geluidssignalen wanneer ze niet bepaald kunnen worden op basis van gebruiksdata. We vergelijken een traditionele aanpak op basis van een *bag-of-words* voorstelling van de geluidssignalen met een diep convolutioneel neuraal netwerk, en evalueren de voorspellingen op kwantitatieve en kwalitatieve wijze op de Million Song Dataset. Hierdoor tonen we aan dat het gebruik van uit geluidssignalen voorspelde latente factoren redelijke aanbevelingen oplevert, en dit ondanks het feit dat er een grote semantische kloof is tussen de eigenschappen van een nummer die de voorkeur van de luisteraars beïnvloeden en het overeenkomstige geluidssignaal. We tonen ook aan dat recent geïntroduceerde deep learning technieken ook goed toepasbaar zijn in de context van muziekaanbeveling: diepe convolutionele neurale netwerken presteren heel wat beter voor deze taak dan de traditionele aanpak.

End-to-end leren van muzikale geluidskenmerken

Inhoudsgebaseerde MIR-taken worden traditioneel opgelost met behulp van manueel ontworpen kenmerken en ondiepe architecturen voor de verwerking

van geluidssignalen. Tegenwoordig is er steeds meer interesse in het gebruik van geleerde kenmerken en diepe architecturen. Deze andere manier van werken vermindert de nood aan domeinkennis en vermijdt zelfs de kost van het manueel ontwerp van goede taakspecifieke kenmerken. Desondanks steunt deze nieuwe aanpak vaak wel nog op zogenaamde *mid-level* voorstellingen van muzikale geluidssignalen, bv. spectrogrammen, in plaats van rauwe, onbewerkte geluidssignalen.

We onderzochten of het mogelijk is om algoritmen voor het automatisch leren van kenmerken rechtstreeks toe te passen op rauwe geluidssignalen. We trainden convolutionele neurale netwerken voor het automatisch taggen van muziek op basis van beide soorten voorstellingen van de invoer, en vergeleken de resultaten. Hoewel netwerken op basis van rauwe audiosignalen het niet beter doen dan netwerken op basis van spectrogrammen, zijn ze wel in staat om op een autonome manier frequentiedecomposities en fase- en translatie-invariante voorstellingen van geluidssignalen te leren.

Summary

Music is increasingly consumed in a digital format. Many people have grown accustomed to buying music in the form of audio files rather than physical media, or streaming music from the internet. This digital distribution and consumption of music has created plenty of opportunities for automation. Browsing and searching large catalogues of music can be facilitated by analyzing the associated metadata as well as the audio content, in order to classify and organize the music in various ways. This benefits both the listeners, by aiding them in maintaining their personal collections, and the artists and music vendors, who can boost their revenue by making it easier for potential customers to find what they are looking for.

Music information retrieval

The field of *music information retrieval (MIR)* concerns itself with the analysis of music and associated metadata through use of computational methods, and combines ideas from musicology, signal processing and machine learning. Researchers in *content-based* MIR are particularly interested in extracting music information from audio waveforms. This is a challenging task: there is quite a large semantic gap between the audio waveform representing a piece of music, and the various high-level aspects of the music that affect our preference and consumption patterns (such as genre, instrumentation, ...).

This thesis

Music audio is inherently hierarchically structured in many different ways. In the research described in this thesis, I have developed methods for content-based MIR which exploit this hierarchical structure of music. For this pur-

pose, I have chiefly made use of techniques from *deep learning*, an emerging subfield of machine learning. The idea of *deep learning* is to build models that represent data at multiple levels of abstraction, and can discover such representations autonomously from the data itself. Deep learning models consist of several layers of processing that form a hierarchy: each subsequent layer extracts a progressively more abstract representation of the input data and builds upon the representation from the previous layer. This model structure is an excellent match for the hierarchical structure of music.

Music classification with a pre-trained convolutional neural network

The *Million Song Dataset* contains audio features and metadata for one million songs. We have built a convolutional network that is trained on this dataset to perform artist recognition, genre recognition and key detection. The network is tailored to summarize the audio features over musically significant timescales. Because labels are scarce, it is infeasible to train the network on all available data in a supervised fashion. We use unsupervised pre-training to be able to harness the entire dataset: we train a convolutional deep belief network on all data, and then use the learnt parameters to initialize a convolutional multilayer perceptron with the same architecture. The MLP is then finetuned on a labeled subset of the data for each task.

Multiscale feature learning for music audio

Content-based music information retrieval tasks are typically solved with a two-stage approach: features are extracted from music audio signals, and are then used as input to a regressor or classifier. These features can be engineered or learnt from data. Although the former approach was dominant in the past, feature learning has started to receive more attention from the MIR community in recent years. There has also been increased interest in multiscale representations of music audio recently. Such representations are more versatile because music audio exhibits structure on multiple timescales, which are relevant for different MIR tasks to varying degrees.

Inspired by recent results in feature learning using very simple and fast algorithms such as K-means to great effect, we propose three different architectures for multiscale audio feature learning using the spherical K-means algorithm. We evaluate them for automatic tagging, similarity metric learning and genre recognition on the Magnatagatune and GTZAN datasets, as well as subsets of the Million Song Dataset.

Deep content-based music recommendation

Automatic music recommendation has become an increasingly relevant problem in recent years, since a lot of music is now sold and consumed digitally. Most recommender systems rely on collaborative filtering. However, this approach suffers from the cold start problem: it fails when no usage data is available, so it is not effective for recommending new and unpopular songs.

We propose to use a latent factor model for recommendation, and predict the latent factors from music audio when they cannot be obtained from usage data. We compare a traditional approach using a bag-of-words representation of the audio signals with deep convolutional neural networks, and evaluate the predictions quantitatively and qualitatively on the Million Song Dataset. We show that using predicted latent factors produces sensible recommendations, despite the fact that there is a large semantic gap between the characteristics of a song that affect user preference and the corresponding audio signal. We also show that recent advances in deep learning translate very well to the music recommendation setting, with deep convolutional neural networks significantly outperforming the traditional approach.

End-to-end learning for music audio

Content-based music information retrieval tasks have traditionally been solved using engineered features and shallow processing architectures. In recent years, there has been increasing interest in using feature learning and deep architectures instead, thus reducing the required engineering effort and the need for prior knowledge. However, this new approach typically still relies on mid-level representations of music audio, e.g. spectrograms, instead of raw audio signals.

We investigate whether it is possible to apply feature learning directly to raw audio signals. We train convolutional neural networks using both approaches and compare their performance on an automatic tagging task. Although they do not outperform a spectrogram-based approach, the networks are able to autonomously discover frequency decompositions from raw audio, as well as phase- and translation-invariant feature representations.

List of abbreviations

ALS	Alternating Least Squares
API	Application Programming Interface
AUC	Area Under the ROC-curve
BoW	Bag-of-words
CD	Contrastive Divergence
CDBN	Convolutional Deep Belief Network
CD-k	k-step Contrastive Divergence
CF	Collaborative Filtering
CNN	Convolutional Neural Network
Convnet	Convolutional Neural Network
CPU	Central Processing Unit
CQT	Constant-Q Transform
CV	Cross-validation
DBM	Deep Boltzmann Machine
DBN	Deep Belief Network
DCT	Discrete Cosine Transform
DNN	Deep Neural Network
FFT	Fast Fourier Transform
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
GTZAN	George Tzanetakis's genre classification dataset
Gyr	Gigayear, 10^9 years
GZ2	Galaxy Zoo 2

LSST	Large Synoptic Survey Telescope
LSTM	Long Short-term Memory
mAP	Mean Average Precision
MF	Matrix Factorization
MFCC	Mel Frequency Cepstral Coefficient
MIDI	Musical Instrument Digital Interface
MIR	Music Information Retrieval
MLP	Multilayer Perceptron
MLR	Metric Learning to Rank
MSD	Million Song Dataset
MSE	Mean Squared Error
NCA	Neighbourhood Components Analysis
NLL	Negative Log Likelihood
NN	Neural Network
PCA	Principal Components Analysis
PCM	Pulse Code Modulation
PMF	Probabilistic Matrix Factorization
PMSC	Principal Mel-Spectrum Component
RBM	Restricted Boltzmann Machine
ReLU	Rectified Linear Unit
RGB	Red, Green, Blue
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
SDSS	Sloan Digital Sky Survey
SGD	Stochastic Gradient Descent
STFT	Short-time Fourier Transform
SVM	Support Vector Machine
t-SNE	t-distributed Stochastic Neighbour Embedding
VAE	Variational Autoencoder
WAVE	Wave audio file format
WMF	Weighted Matrix Factorization
WPE	Weighted Prediction Error

Contents

1 Introduction

Music is increasingly consumed in a digital format. Many people have grown accustomed to buying music in the form of audio files rather than physical media, or streaming music from the internet. This digital distribution and consumption of music has created plenty of opportunities for automation. Browsing and searching large catalogues of music can be facilitated by analyzing the associated metadata as well as the audio content, in order to classify and organize the music in various ways. This benefits both the listeners, by aiding them in maintaining their personal collections, and the artists and music vendors, who can boost their revenue by making it easier for potential customers to find what they are looking for.

The field of *music information retrieval (MIR)* concerns itself with the analysis of music and associated metadata through use of computational methods, and combines ideas from musicology, signal processing and machine learning. Researchers in *content-based* MIR are particularly interested in extracting music information from audio waveforms. This is a challenging task: there is quite a large semantic gap between the audio waveform representing a piece of music, and the various high-level aspects of the music that affect our preference and consumption patterns. These aspects are precisely the ones that are of interest when people search music catalogues, so being able to extract them automatically is valuable.

Music audio is inherently hierarchically structured in many different ways. Most contemporary music exhibits harmonic structure based on chord progressions, which are temporal sequences of chords. Chords are composed of pitches, which in turn consist of various frequencies: fundamental frequencies and their multiples. Music also features a hierarchical temporal structure in the form of rhythm. On a coarser timescale, we can group note sequences into motifs, themes and phrases. Classical music often adhered quite rigorously to *musical forms* (e.g. the sonata form), which describe

the large-scale temporal structure of a piece in detail. Contemporary music often exhibits a verse-chorus-verse structure.

In the research described in this thesis, I have developed methods for content-based MIR which exploit this hierarchical structure of music. For this purpose, I have chiefly made use of techniques from *deep learning*, an emerging subfield of machine learning. The idea of *deep learning* is to build models that represent data at multiple levels of abstraction, and can discover such representations autonomously from the data itself [8]. Deep learning models consist of several layers of processing that form a hierarchy: each subsequent layer extracts a progressively more abstract representation of the input data and builds upon the representation from the previous layer. This model structure is an excellent match for the hierarchical structure of music.

This chapter is intended to provide those who are unfamiliar with the subject matter with an overview of music information retrieval, machine learning and deep learning, and to describe the context in which this research took place. It concludes with an outline of the thesis and an overview of research contributions, as well as a list of publications.

1.1 Music information retrieval

Music information retrieval is a very broad field at the crossroads of various domains. It brings together researchers from very different backgrounds including musicology, signal processing and machine learning, all of whom investigate how higher-level representations and information can be extracted from music. Their motivations are very diverse. For example, musicologists may use MIR techniques to extract musically interesting properties from large collections, or they may use them to support experiments with musicians or listeners. Developers might want to build tools for musicians or for music education, such as automatic transcription or score-following software. Yet others are interested in automatically annotating or recommending music, in order to help listeners and musicians find each other. The work described in this thesis falls squarely within this last category.

MIR tasks can be categorized along several axes:

- **data sources and types**: one can extract musical information from raw audio signals, symbolic representations of music (such as MIDI files or images of sheet music), associated metadata (composer, artist, date of publication, or manually annotated information about the audio content such as genre labels and tags), lyrics, or textual data mined from the web (blog posts, reviews, mentions on social media). *Content-*

based MIR refers to those tasks where the data source is the music itself, either in the form of raw audio or a symbolic representation.

- **specificity**: some MIR tasks involve identifying a very specific and/or uncommon feature of a piece of music out of a large number of possibilities, such as the name of a song or the performing artist. Examples of such tasks are artist identification and cover song detection. Examples of less specific tasks (involving much broader categories and more common features) are genre classification and instrument classification.

- **abstraction level**: tasks which extract information that is semantically close to the format of the input data are said to be *low-level*. Examples of such tasks are music transcription (i.e. creating a score or MIDI representation from an audio file) and score following (automatically showing an artist the right page of the score and their position within it while they are playing). In *high-level* tasks, like genre classification and recommendation, the information to be extracted is relatively far removed from the input representation in terms of abstraction level. The abstraction level corresponds to the size of the *semantic gap* between the input representation and the information we wish to extract from it.

- **analysis vs. synthesis**: usually the goal of a MIR task is to extract musical information (analysis), but it could also be to generate music (synthesis). This way one could create intelligent instruments that are able to play along with an artist autonomously, for example.

The work described in this thesis focuses on content-based MIR (using musical audio signals in particular), and on tasks with low specificity across a range of abstraction levels.

Audio-based MIR is a relatively small field of research, especially compared to the closely related field of speech processing and recognition. An important reason for this is that it is difficult to obtain large enough datasets due to licensing issues. As a result, researchers are dissuaded to share large collections of music audio, and prefer to assemble and use their own personal datasets. This hampers progress because research results are difficult to reproduce and compare, and collecting and annotating data takes up valuable research time.

A lot of work in audio-based MIR has been inspired by related work in speech recognition: many models and feature representations of audio signals used in speech processing have been repurposed by MIR researchers. The most notable example is the ubiquity of *Mel Frequency Cepstral Coefficients* (MFCCs) which are features extracted from audio signals that capture the short-term power spectrum. These features were originally designed for

speech, but have since been shown to be very versatile in the context of MIR as well. Researchers have repurposed them for use in perceptual audio similarity measures, instrument classification and genre classification, among others.

1.1.1 The semantic gap in music

Content-based MIR tasks can be challenging, because there is typically a large *semantic gap* between the musical properties of interest and the corresponding audio signals. Some properties are easier to extract than others, and this is usually tied to their abstraction level. For example, determining the instruments used in a recording of a piece of music is fairly easy by modern standards, because it requires detecting the presence of certain timbres in the audio signal, which correspond quite well to spectral envelopes. Classifying audio recordings by genre is a bit more difficult, because there are various patterns in the audio signal at various different timescales for which the probability of occurrence depends on the genre of the music [147]. This is because the genre will affect e.g. the instrumentation, which is easy to detect at a very fine-grained timescale, but also the musical form (e.g. verse-chorus-verse structure), which can only be detected at a very coarse timescale. In addition, the concept of a genre is less well-defined than that of an instrument class, so there is more ambiguity in the class labels.

Some properties of music will affect the audio signal in ways that are infeasible to extract automatically: the genre of the music will affect its lyrical content, but extracting lyrics from audio signals and analyzing their underlying meaning is quite challenging and probably not worth the considerable engineering effort involved. Yet others are often not present in the audio signal at all, such as the geographical location of the performing artist[1].

Bridging the semantic gap in an automated fashion can be challenging, and this problem has often been solved in the past by including humans in the loop. The Music Genome Project[2], for example, is a collection of annotations for about 1 million songs with over 450 different attributes. The annotations were obtained by having music experts listen to and manually label each of the songs. This massive undertaking was started in 1999 and is ongoing today. The data from the Music Genome Project is not publicly available; it is currently used to power Pandora[3]'s music recommendation and radio programming algorithms.

Unfortunately this approach is quite costly to scale, and many of Pan-

[1] this property is very relevant in music recommendation, for example, because listeners are more likely to listen to artists from their own area.

[2] https://www.pandora.com/about/mgp

[3] http://www.pandora.com/

dora's competitors boast music collections that are an order of magnitude larger (e.g. Spotify's catalog features over 30 million songs). As a result, they have to use a different approach to recommend music. Automating the annotation process as much as possible makes it feasible to annotate larger collections, and this is where content-based MIR approaches can be especially useful.

1.1.2 Musical audio signal representations

Audio signals represent pressure waves: moving patterns of low and high pressure. They are continuous in both time and amplitude. To store them in digital form, however, the signals need to be quantized in both of these dimensions. This is referred to as *pulse-code modulation* (PCM). The result is a sequence of integers[4] that represent an approximation of the original waveform. Although quantization of a waveform in time (i.e. *sampling*) need not necessarily lead to a loss of information for bandlimited signals, quantization in amplitude always does.

The commonly used WAVE file format simply stores these sequences of numbers unmodified. Other audio file formats, such as MP3, compress the sequences with lossy schemes that remove information the human ear is insensitive to, to reduce the amount of required storage space.

In the context of MIR, however, so-called *mid-level representations* are often extracted from digital audio signals and operated on instead [106]. They typically represent sound in a way that matches more closely how humans perceive it: as varying patterns of different frequencies over time. They are referred to as such because they hold the middle between low-level representations (i.e. raw audio signals) and high-level symbolic representations of music.

The most commonly used mid-level representations are *time-frequency representations*: instead of describing the amplitude of the signal as it varies over time, they describe the amplitudes or energies of the signal in various different *frequency bands*, as they vary over time on a much coarser timescale [161]. In effect, these representations trade temporal resolution for frequency resolution[5].

Many commonly used time-frequency representations are based on *spectrograms*. A spectrogram of a signal can be obtained by computing the short-time Fourier transform (STFT) of short windows that usually overlap partially. From these transformed windows, power spectra can be obtained

[4] Or multiple sequences of integers, in the case of audio signals with multiple channels.

[5] It is not possible for a representation to have the highest resolution in both time and frequency; one has to be traded for the other. This is a form of the Heisenberg uncertainty principle.

that indicate the energy in various frequency bands within each window. Each of these constitutes a *frame*. The successive frames are concatenated into a matrix to form the spectrogram.

Spectrograms are often post-processed further to better match the properties of human perception. The energy values in the spectrogram are often converted to a logarithmic scale, to match our logarithmic perception of loudness. The frequency axis, which is linear by default as a result of the STFT, is often converted to another scale, such as the mel scale [140] (giving rise to *mel spectrograms*, which are commonly used in speech processing) or a logarithmic scale (resulting in *constant-Q* or *log-frequency spectrograms*, where intervals between notes are equidistant across the frequency axis [126]). It is also possible to obtain time-frequency representations using custom filter banks (this approach is sometimes used to obtain constant-Q spectrograms as well) or wavelet transforms.

High-level representations, such as mel-frequency cepstral coefficients (MFCCs) or chroma representations (which represent only pitch information and remove any timbral variation) are also commonly used [107]. I have mostly avoided using them in this work, except for some baseline experiments, seeing as the goal was to learn useful features from data instead.

1.1.3 Tasks of interest

The primary goals of the work described in this thesis are to facilitate searching, sorting and browsing through large collections of musical audio, as well as generating recommendations from them. Consequently, it covers a subset of audio-based MIR tasks that are potentially useful for the automatic annotation and recommendation of music at the song level. We will look at three types of tasks in particular: *music classification*, *automatic tagging* and *music recommendation*. For a more complete overview of the field of MIR and its applications, we refer to Müller [107].

Music classification

Classification is the task of categorizing examples into a set of classes by assigning one or more class labels to each of them. These classes are typically non-overlapping, but don't necessarily have to be. Classification can be carried out manually, but increasingly many classification problems are partially or fully automated through use of *machine learning* (see Section 1.2).

Music can be classified according to many different taxonomies on various different levels of abstraction. Popular classification-based MIR problems are instrument, genre and mood classification. Artist identification can also be

cast as a classification problem by creating a separate class for each artist, provided that all artists are known to come from a given set.

Instrument classification: this is regarded as a less challenging problem in audio-based MIR, because the way different instruments manifest themselves in an audio waveform is fairly straightforward: they affect its spectral envelope, which we perceive as the *timbre*.

Genre classification: the concept of a 'genre' is somewhat ill-defined, and will typically manifest itself in an audio waveform in many different ways (instrumentation, rhythm, melody, ...) and at many different time scales. This makes genre classification a more difficult problem than instrument classification. It has nevertheless been a popular MIR task to study, in no small part due to the public availability of the GTZAN genre classification dataset [147]. This dataset, introduced in 2002, features 1000 audio clips divided into 10 genres. As mentioned before, audio-based MIR datasets are hard to come by, and as a result it has been used in a large body of work [143].

Mood classification: the mood of a piece of music is somewhat subjective and arguably even more ambiguous than its genre, so this problem is even more ill-defined. Nevertheless, there is a lot of interest in this task because mood information can be useful for making context-dependent recommendations.

Artist identification: this problem is usually studied on a fairly small scale, where treating it as a classification problem is feasible.

Other classification tasks: other tasks of interesting include identifying the key (i.e. the tonic note and chord) or the tempo of a piece of music.

Classification is useful in the context of making large collections of music easily browsable and searchable, because the assigned class labels can be exposed to the user directly and used as filters. But it is also useful indirectly, for music recommendation for example: we can determine the affinity of a given user for each label based on their prior listening habits, and then find songs to recommend by selecting songs from the classes for which the user has a high affinity. Pandora's radio programming algorithms are based on this approach, using manual classifications obtained from experts (see Section 1.1.1).

Automatic tagging

Tags are keywords or short descriptions that can be associated with various types of digital content, such as images, news articles or web pages. They are also popular for categorizing and organizing music. In many cases, tags can be associated with content by users, and the tags do not come from a

prespecified taxonomy: the users can use any word or combination of words as a tag. A good example is the last.fm[6] platform, where all songs, artists and albums can be tagged by registered users. The most popular tags for each content item are displayed on the associated web pages.

Because tags are arbitrarily chosen by many different users, they constitute a fairly noisy form of metadata. Different users may use synonyms or different spellings for the same word (e.g. 'favorite', 'favourite'), leading to redundancy. Many users use the tagging system to create categories that are useful only to themselves (e.g. the 'seen live' or 'favorite' tags on last.fm). Usually, tags form a flat taxonomy, without any form of hierarchy. In some cases, tags may instead be assigned only by trained experts, and from a limited predefined taxonomy (e.g. the Music Genome Project). In this case, the tags are usually much more consistent and reliable.

The goal of *automatic tagging* is to automatically assign relevant tags to pieces of content. Automatic tagging systems are usually built to reproduce tag data obtained from users, so that they can be trained on this data. This means it is useful to treat it as a problem category of its own. It is also possible to cast automatic tagging as a classification problem. However, it has several properties that set it apart from typical classification problems, especially in a music context:

- Tag prediction is a *multi-label classification* task: each song can be associated with multiple tags, so the classes are not disjoint.
- There are large numbers of tags; orders of magnitude more than there are classes in typical classification problems.
- Tags are weak labels: if a song is not associated with a particular tag, the tag may still be applicable to the song. In other words, some positive labels are missing. This also has implications for the metrics we use to measure the performance of automatic tagging systems.
- The labels are redundant: a lot of tags are correlated, or have overlapping meanings. For example, songs tagged with *disco* are more likely to also be tagged with *80's*. Misspellings and synonyms exacerbate this problem.
- The labels are very sparse: most tags only apply to a small subset of songs.

Nevertheless, for practical automatic tagging problems with limited taxonomies, creating a binary classifier for each tag is often sufficient.

[6] http://www.last.fm/

Music recommendation

The goal of music recommendation is to recommend new artists and songs to listen to based on users' preferences. These preferences can be determined in many different ways and from various different data sources. Consequently, there is a large variety of approaches to building recommender systems for music. Recommender systems in general have become very common in recent years and are applied in a variety of different settings. Besides music, recommender systems exist for movies, books, products (e.g. in online shops) and people (online dating).

Although recommender systems have been studied extensively, the problem of music recommendation in particular is complicated by the sheer variety of different styles and genres, as well as social and geographic factors that influence listener preferences. The number of different items that can be recommended is very large, especially when recommending individual songs. This number can be reduced by recommending albums or artists instead, but this is not always compatible with the intended use of the system (e.g. automatic playlist generation), and it disregards the fact that the repertoire of an artist is rarely homogenous: listeners may enjoy particular songs more than others.

The most popular approach to build a recommender system, both in academia and industry, is using *collaborative filtering*. This technique can be used to extract preference information for users from their prior consumption patterns. As a result, it is *content-agnostic*: it can be applied to any type of recommendation problem. It is based on the assumption that we can predict which items a given user will like based on the preference of other users whose consumption patterns are similar to theirs. Collaborative filtering works very well for music. Preference information can be obtained by having users rate songs or mark them as favourites (*explicit feedback*), or by analyzing their listening patterns and other behaviours (*implicit feedback*). The latter approach is more common in practice because it does not require any effort on the part of the user.

It is also possible to build recommender systems that use information about the users and the items. For example, we can use demographic information about users, as this almost certainly affects their taste in music. We can also use metadata about songs and artists (geographical location, year of release, genre labels, tags), or the audio content of the songs. We will refer to this as *content-based* recommendation. This approach is not as popular, because it is typically more complex than collaborative filtering and requires more data to be available. Collaborative filtering also tends to outperform content-based approaches in many situations.

Nevertheless, collaborative filtering has certain problems that content-

based approaches do not suffer from. The so-called *cold start problem* is probably the most important: it is impossible to recommend new and unpopular items using collaborative filtering, because not enough consumption information is available to analyse. For the same reason, it is also impossible to make recommendations for new users. This is why studying content-based recommendation is worthwhile, and especially for music, where the cold-start problem is exacerbated because so many niche genres and relatively unpopular artists and songs exist.

A lot of other MIR tasks can indirectly be useful for music recommendation: for example, tags predicted by an automatic tagging system can be used to find good recommendations. However, we can also study music recommendation as a task in its own right and attempt to build an end-to-end solution for this problem. Note that the problems of music recommendation and automatic tagging are closely related: just like tag prediction, user preference prediction can be seen as a multi-label classification task. Users who have not listened to a given song might still like it if they heard it, so a lot of positive information is missing here as well. There is also a lot of redundancy because many users have similar tastes, and the consumption patterns tend to be very sparse because users only listen to a very small subset of songs.

1.2 Machine learning

Machine learning is the study of systems that are able to learn from examples without being explicitly programmed. This can be achieved by constructing a parameterised model and then finding the optimal set of parameters by minimizing an objective function that measures how well the model fits the examples (i.e. an error measure). The goal is to make the model fit the example data well, so it can make predictions about it, classify it, detect anomalies or generate new examples.

Being able to solve prediction and classification tasks without having to explicitly write a program to do so is valuable, because many such tasks are difficult to capture in an algorithm. This is especially true for certain tasks that are very easy to do for humans, such as identifying objects in an image, or understanding speech. Machine learning makes it possible for computers to perform these tasks without requiring a manually engineered solution. This reduces development costs and ultimately leads to better performance and more robust solutions.

Machine learning can be cast as an optimisation problem: given a *dataset* consisting of *training examples*, a machine learning model is *trained* by op-

timising an objective function. However, it differs from typical optimisation problems in a key point: the goal is to achieve good performance on new, previously unseen data, instead of maximising performance on the available training data. This is an extremely important distinction: if the parameters of a model are simply optimised to minimise the training objective, this will likely result in *overfitting*. The model will exhibit good performance on the training data, but poor generalisation performance.

Overfitting is a consequence of the fact that training data for a machine learning model is usually noisy. Powerful models may be able to fit this noise exactly, allowing them to memorise each training example. To prevent this from happening, the complexity of the model needs to be controlled by *regularisation*.

Many practical problems can be solved by function approximation. In this case, the training examples for a machine learning model would be pairs (x, t), where x represents an example and t represents the desired output of the function for this example. The output of the model $y = f(x)$ should then approximate t as well as possible. This setting is referred to as *supervised learning*. Examples of supervised learning include classification, regression (predicting one or more continuous values) and ranking.

In the case of *unsupervised learning*, the model should instead discover structure in the training data. Examples of unsupervised learning are clustering, dimensionality reduction (mapping the examples into a lower-dimensional space), feature learning (learning useful representations for use in supervised learning problems) and density modeling (approximating the probability distribution of the training data).

This section is intended to provide an overview of the field of machine learning, with a focus on the concepts and models that I have used in my research.

1.2.1 Supervised learning

Let the vector $\mathbf{x}_n$ be a data point from a dataset $\mathcal{D}$ consisting of N examples, and $\mathbf{t}_n$ its associated target vector:

$$\mathcal{D} = \{(\mathbf{x}_n, \mathbf{t}_n), n = 0, \ldots, N-1\}.$$

We will represent both the data point and the target as vectors for convenience, but they could have any dimensionality. Both the data points and targets can be discrete or continuous. Let $f(\mathbf{x}|\boldsymbol{\theta})$ be a parameterised function representing the model, with parameter vector $\boldsymbol{\theta}$. Once the model is

trained, we can obtain a prediction $\mathbf{y}_n$ for each data point $\mathbf{x}_n$:

$$\mathbf{y}_n = f(\mathbf{x}_n|\boldsymbol{\theta}).$$

The goal of supervised learning is to make the predictions $\mathbf{y}_n$ approximate the targets $\mathbf{t}_n$ as accurately as possible. We will need to define an objective function $J(\mathcal{D}, \boldsymbol{\theta})$ that measures the quality of the approximation across all $(\mathbf{x}_n, \mathbf{t}_n) \in \mathcal{D}$. Finally, we need an algorithm to optimise $J(\mathcal{D}, \boldsymbol{\theta})$.

Regression

For a regression problem, $\mathbf{t}_n$ is a continuous-valued scalar or vector. For notational convenience, we will consider the case of scalar targets t_n in what follows. As a result, the model predictions y_n will also be scalar-valued. A common choice for the objective function in this case is the *mean squared error* (MSE):

$$J(\mathcal{D}, \boldsymbol{\theta}) = \frac{1}{N} \sum_{n=0}^{N-1} (t_n - y_n)^2 = \frac{1}{N} \sum_{n=0}^{N-1} (t_n - f(\mathbf{x}_n|\boldsymbol{\theta}))^2 .$$

This is the average across all data points of the squared difference between the target value and the predicted value. Of course, other types of objective functions are also possible. Because this objective function measures the prediction error, it should be minimised with respect to the model parameters $\boldsymbol{\theta}$ to find the optimal solution:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} J(\mathcal{D}, \boldsymbol{\theta}).$$

Classification

In a classification setting, each data point $\mathbf{x}_n$ is understood to belong to a subset of K classes, and the goal for the model is to be able to identify which of these classes each data point belongs to. Typically, the classes will be mutually exclusive, so each data point belongs to one class only, but they do not have to be. We will assume that this is the case here. The K classes can be represented by indices ranging from 0 to $K - 1$. We will use such indices for the targets t_n and the corresponding predictions y_n, which are both discrete in this case.

A classification model assigns a score $s_{nk}, k = 0, \ldots, K - 1$ to each class, given a data point $\mathbf{x}_n$. The predicted class is then the one with the highest score:

$$y_n = \arg\max_k s_{nk}.$$

Classification is often approached using a probabilistic interpretation. The scores s_{nk} are then constrained to form a discrete probability distribution across the classes:

$$0 \leq s_{nk} \leq 1 \quad \forall n, k,$$

$$\sum_{k=0}^{K-1} s_{nk} = 1 \quad \forall n,$$

and we define $p(y_n = k) = s_{nk}$. We can then use the *conditional log-likelihood* $p(y|\mathbf{x})$ of the model predictions given the data points as the objective function:

$$J(\mathcal{D}, \boldsymbol{\theta}) = \sum_{n=0}^{N-1} \log p(y_n = t_n | \mathbf{x}_n).$$

This objective should be maximised, because we wish to maximize the likelihood of the data under the model. By convention, a minus sign is often added in front to get the *negative log-likelihood* (NLL), turning it into a minimisation problem.

The scores can be constrained to form a probability distribution by using the so-called *softmax* function in the model:

$$s_{nk} = \frac{\exp(a_{nk})}{\sum_{k=0}^{K-1} \exp(a_{nk})},$$

where a_{nk} are real numbers that come from the model.

1.2.2 Unsupervised learning

In the case of unsupervised learning, the dataset $\mathcal{D}$ consists only of the data points $\mathbf{x}_n$, because there are no associated target vectors:

$$\mathcal{D} = \{\mathbf{x}_n, n = 0, \ldots, N-1\}.$$

Because there are no targets, we cannot define the objective function $J(\mathcal{D}, \boldsymbol{\theta})$ in terms of how well the targets are approximated by the model. Instead, we have to specify it directly in terms of the data points. Like before, it should measure how well the model fits the data. We will also need an algorithm to optimise $J(\mathcal{D}, \boldsymbol{\theta})$.

Density modeling

The goal of density modeling is to model the distribution $p(\mathbf{x})$ of the data over the input space. The objective is then to maximise the log-likelihood

of the data under the model:

$$J(\mathcal{D}, \boldsymbol{\theta}) = \sum_{n=0}^{N-1} \log p(\mathbf{x}_n).$$

Note that this is now expressed in terms of the distribution $p(\mathbf{x})$, and not in terms of the conditional distribution $p(\mathbf{t}|\mathbf{x})$ as was the case for classification.

Density modeling is arguably one of the most versatile forms of unsupervised learning. Many types of models allow for the likelihood of new data points to be estimated, and for plausible samples of new datapoints to be drawn.

Clustering

Clustering entails finding groups of similar data points in a set. Some clustering algorithms require the number of groups to be specified on beforehand, but others do not. The objective function associated with clustering usually rewards similarity between data points that are added to the same cluster and penalizes dissimilarity within each cluster. Additionally it may also penalize similarity between data points from different clusters.

Similarity between data points can be measured as the inverse of the distance between them in the input space. The objective can then be formulated as a minimization of the distance between data points within clusters, while simultaneously maximizing the distance between data points from different clusters. Many clustering algorithms can be paired with different distance metrics to get different results.

Dimensionality reduction

Dimensionality reduction is the task of finding a low-dimensional representation of high-dimensional data that retains as much valuable information as possible. Low-dimensional representations are valuable because high-dimensional data is more difficult to store and process, and many machine learning algorithms scale very poorly as the dimensionality of the data increases (the so-called *curse of dimensionality*). Furthermore, they often make the data easier to interpret for humans.

1.2.3 Other learning paradigms

Although supervised and unsupervised learning are the most commonly studied machine learning paradigms, others exist.

Semi-supervised learning: supervised learning requires the availability of a target vector for each training example. For many practical problems, the training examples themselves are much easier to come by than the associated target vectors (for example because they require manual annotation by humans). Semi-supervised learning allows for data with missing target vectors to be harnessed. Models can still benefit from this data because it contains more information about the distribution of the input. Semi-supervised learning combines ideas from unsupervised and supervised learning to allow for both types of data to be exploited.

Transduction: transduction is a particular form of semi-supervised learning where the data points that the model will have to generate predictions for are already known on beforehand. These data points can also be learnt from in a semi-supervised fashion.

Transfer learning: when several tasks are related, a model trained for one of the tasks may extract knowledge that is also useful for the other tasks. The idea of transfer learning is to transfer knowledge from one task to another, for example by reusing learned representations, or even reusing model parameters directly.

Reinforcement learning: in reinforcement learning, agents learn which actions to take in response to their environment, in order to maximize some form of reward. This is a much more difficult problem than supervised learning, because it is not possible to measure the effect of each action directly - the only information available to the agent to learn from is the eventual cumulative reward resulting from the sequence of actions it took. Reinforcement learning has many applications in robotics.

1.2.4 Underfitting and overfitting

Although training a machine learning model involves fitting its parameters to a set of training examples, the ultimate goal is to achieve good performance on previously unseen examples instead, i.e. good *generalisation performance*. These objectives are not completely aligned: if the model is very powerful, it is possible to fit the training data exactly while performing no better than chance on unseen data. This phenomenon is called *overfitting*. Conversely, if the model is not powerful enough, it might not be able to perform well even on the training data. This situation is referred to as *underfitting*.

Both problems can be demonstrated by considering the task of fitting a polynomial curve to a set of points in a plane. Figure 1.1 shows a function from which a couple of noisy input-output pairs were obtained by adding Gaussian noise with a small standard deviation to the output values. Subsequently, three polynomial curves were fit to these points, of degrees 1, 3 and 9 respectively. Higher-degree polynomial curves have more degrees of

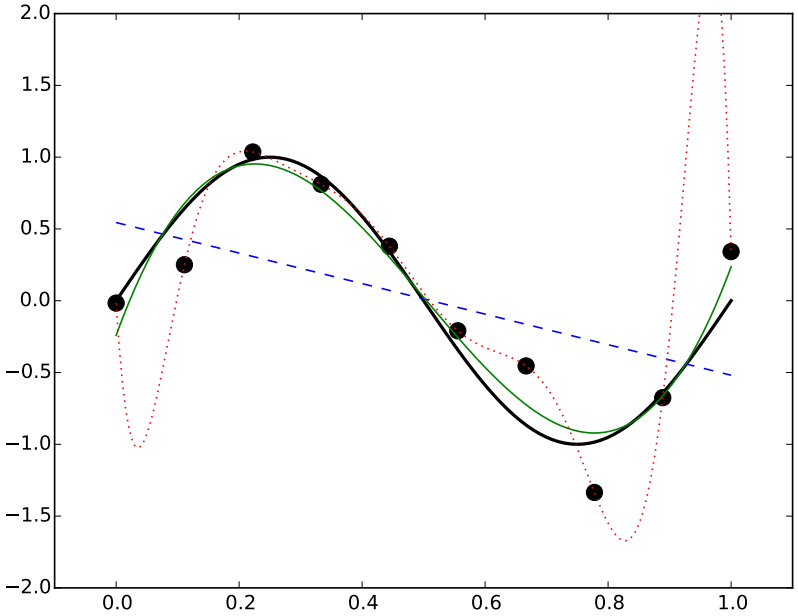


Figure 1.1: Polynomial curve fitting. The original function and some noisy samples from it are shown in black (thick line). Three different polynomial approximations are shown: degree 1 (blue dashed line), degree 3 (green solid line) and degree 10 (red dotted line).

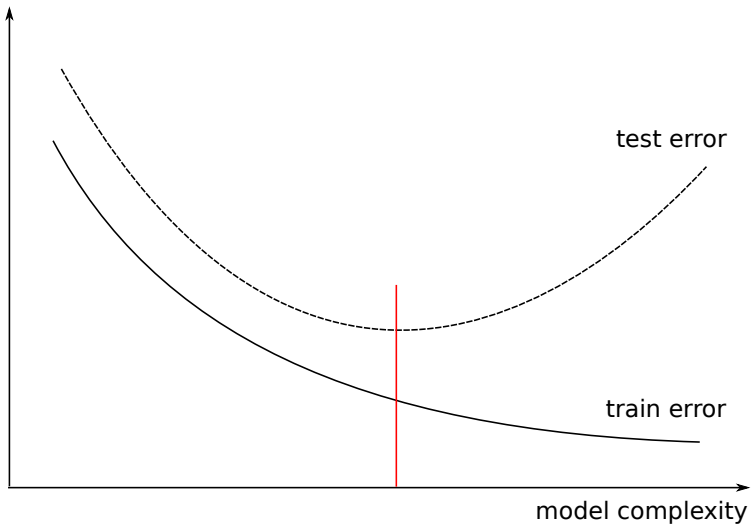


Figure 1.2: Schematic representation of train and test error of a model in function of model complexity. The optimal model complexity is marked by a vertical red line.

freedom: the higher the degree, the more coefficients the polynomial has. Viewed as models, one could say that higher-degree polynomial curves have a higher *modeling capacity*.

As the figure shows, the curve of degree 1 (a line) does not fit the data very well. This is an example of *underfitting*, because a degree 1 polynomial curve clearly cannot approximate the original function well under any circumstances. The curve of degree 3 matches the original function remarkably well, despite the fact that we added Gaussian noise to the sampled points. The curve of degree 9 goes through all of the points[7], but compared to the original function it is a poor fit. This is an example of *overfitting*: the model is actually too powerful and is able to fit the noise, resulting in poor generalisation.

The polynomial curve fitting example is covered in great detail in Chapter 1 of Bishop [19].

Figure 1.2 shows a schematic representation of the train error (modeling error on the training examples) and the test error (modeling error on unseen data) of a model in function of its complexity. For low-complexity models, both the train and test error will be high (underfitting). For high-complexity models, the train error will be low, but the test error can still be high

[7]This is a property of polynomials: for a set of N points, exactly one degree-$N-1$ polynomial can be found that goes through all of them.

(overfitting). There is an optimal model complexity where the minimal test error is achieved, marked by a vertical red line on the figure.

It is worth noting that underfitting and overfitting are not binary phenomena, in the sense that different levels of underfitting and overfitting may occur. In hierarchical models such as the ones described later, both phenomena may even co-occur within different layers of the same model.

1.2.5 Model validation

Because we are not interested in the performance of a model on training data, but rather how it generalises to new, previously unseen data, it is important to *validate* models on a holdout dataset. The training data and this holdout set are typically referred to as the *training set* and the *validation set* respectively.

Apart from trainable parameters, many models have several *hyperparameters* that influence the model or the training procedure. Finding optimal values for these parameters is also done by determining how each parameter choice affects the performance on the validation set. This is especially important because some hyperparameters affect the generalisation capabilities of the model, so it is not possible to measure their influence on the training set at all.

Finally, another holdout set is usually reserved to evaluate the model performance after all parameters and hyperparameters have been optimised: the *test set*. This is done to obtain an unbiased estimate of the real-world performance of the model. This three-way split (training, validation and test sets) is crucial to be able to compare models in an unbiased manner.

Since data is often scarce, splitting it into three sets and only training on one of them is not always feasible - it may have a detrimental effect on performance because it reduces the amount of data available for training. Instead, a common approach is to split off only the test data, and then iteratively split the remainder into different training and validation sets several times. Each split is called a *fold*. If there are K such folds, this procedure is termed *K-fold crossvalidation*. The resulting performance metrics are then averaged across folds to be able to compare different models. Sometimes *nested* crossvalidation is performed: the procedure is also applied recursively, once to split off a test set and once to split off a validation set.

1.2.6 Regularisation

To mitigate overfitting, a reduction of model complexity is typically in order. Model complexity stems from the number of learnable parameters, but also from how these parameters are used in the model. As a result, reducing

model complexity does not necessarily imply a reduction in the number of parameters.

In many types of models (including linear models), overfitting is often associated with large parameter values[8]. This can be mitigated by adding a term to the objective function that penalizes large parameter values. A very commonly used regularisation method is to add the sum of squared parameter values to the objective:

$$J'(\mathcal{D}, \boldsymbol{\theta}) = J(\mathcal{D}, \boldsymbol{\theta}) + \lambda \cdot ||\boldsymbol{\theta}||^2.$$

This is called *L2 regularisation*, after the L2 norm. The *regularisation parameter* λ controls the strength of the regularisation. The larger its value, the stronger the resulting regularisation effect will be. Other types of penalties such as the L1 norm can also be used.

Another common regularisation approach is to inject noise into the model during training, to prevent the model from fitting the intrinsic noise in the training data.

1.3 Deep learning and neural networks

The idea of *deep learning* is to build models that represent data at multiple levels of abstraction, and can discover accurate representations autonomously from the data itself [8]. Deep learning models consist of several layers of processing that form a hierarchy: each subsequent layer extracts a progressively more abstract representation of the input data and builds upon the representation from the previous layer, typically by computing a non-linear transformation of its input. The parameters of these transformations are optimized by *training* the model on a dataset.

1.3.1 Neural networks

A *feed-forward neural network* is an example of such a model, where each layer consists of a number of units (or neurons) that compute a weighted linear combination of the layer input, followed by an elementwise non-linearity. These weights constitute the model parameters. Let the vector $\mathbf{x}_{n-1}$ be the input to layer n, $\mathbf{W}_n$ be a matrix of weights, and $\mathbf{b}_n$ be a vector of biases. Then the output of layer n can be represented as the vector

$$\mathbf{x}_n = f(\mathbf{W}_n \mathbf{x}_{n-1} + \mathbf{b}_n), \tag{1.1}$$

[8] Large in magnitude, so either very positive or very negative values.

where f is the *activation function*, an elementwise non-linear function. Common choices for the activation function are linear rectification $[f(x) = \max(x, 0)]$, which gives rise to *rectified linear units* [ReLUs; 110], or a sigmoidal function $[f(x) = (1 + e^{-x})^{-1}$ or $f(x) = \tanh(x)]$. Another possibility is to compute the maximum across several linear combinations of the input, which gives rise to *maxout units* [49]. We will consider a network with N layers. The network input is represented by $\mathbf{x}_0$, and its output by $\mathbf{x}_N$.

A schematic representation of a feed-forward neural network is shown in Figure 1.3. The network computes a function of the input $\mathbf{x}_0$. The output $\mathbf{x}_N$ of this function is a prediction of one or more quantities of interest. We will use $\mathbf{t}$ to represent the desired output (target) corresponding to the network input $\mathbf{x}_0$. The topmost layer of the network is referred to as the *output layer*. All the other layers below it are *hidden layers*.

1.3.2 Training

During training, the parameters of all layers of the network are jointly optimized to make the output $\mathbf{x}_N$ approximate the desired output $\mathbf{t}$ as closely as possible. We quantify the prediction error using an error measure $e(\mathbf{x}_N, \mathbf{t})$. As a result, the hidden layers will learn to produce representations of the input data that are useful for the task at hand, and the output layer will learn to predict the desired output from these representations.

To determine how the parameters should be changed to reduce the prediction error across the dataset, we can use *gradient descent*: the gradient of $e(\mathbf{x}_N, \mathbf{t})$ is computed with respect to the model parameters $\mathbf{W}_n$, $\mathbf{b}_n$ for $n = 1 \ldots N$. The parameter values of each layer are then modified by repeatedly taking small steps in the direction opposite to the gradient:

$$\mathbf{W}_n \leftarrow \mathbf{W}_n - \eta \frac{\partial e(\mathbf{x}_N, \mathbf{t})}{\partial \mathbf{W}_n}, \tag{1.2}$$

$$\mathbf{b}_n \leftarrow \mathbf{b}_n - \eta \frac{\partial e(\mathbf{x}_N, \mathbf{t})}{\partial \mathbf{b}_n}. \tag{1.3}$$

Here, η is the *learning rate*, a hyperparameter controlling the step size.

1.3.3 Deep neural networks

Traditionally, models with many non-linear layers of processing have not been commonly used because they were difficult to train: gradient information would vanish as it propagated through the layers, making it difficult to learn the parameters of lower layers [62]. Practical applications of neural

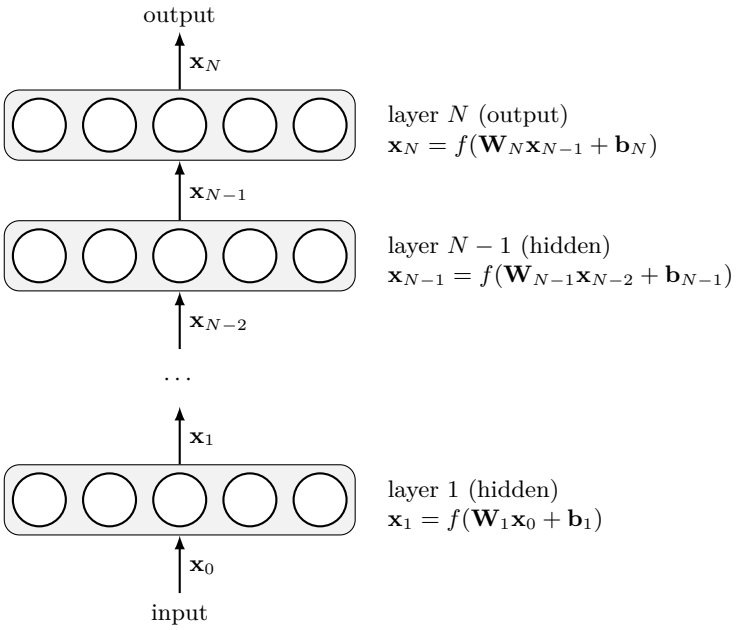


Figure 1.3: Schematic representation of a feed-forward neural network with N layers.

networks were limited to models with one or two hidden layers. Since 2006, the invention of several new techniques, along with a significant increase in available computing power, have made this task much more feasible.

Initially *unsupervised pre-training* was proposed as a method to facilitate training deeper networks [61]. Single-layer unsupervised models (such as restricted Boltzmann machines or auto-encoders [8]) are stacked on top of each other and trained, and the learned parameters of these models are then used to initialize the parameters of a deep neural network. These are then fine-tuned using standard gradient descent. This initialization scheme makes it possible to largely avoid the vanishing gradient problem. Nair and Hinton [110] and Glorot et al. [45] proposed the use of rectified linear units (ReLUs) in deep neural networks. By replacing traditional activation functions with linear rectification, the vanishing gradient problem was significantly reduced. This also makes pre-training unnecessary in most cases.

The introduction of dropout regularization [57, 138] has made it possible to train larger networks with many more parameters. Dropout is a regularization method that can be applied to a layer n by randomly removing the output values of the previous layer $n-1$ (setting them to zero) with probability p. Typically p is chosen to be 0.5. The remaining values are rescaled by a factor of $(1-p)^{-1}$ to preserve the scale of the total input to each unit in layer n. For each training example that is presented to the network, a different subset of values is removed. During evaluation, no values are removed and no rescaling is performed.

Dropout is an effective regularizer because it prevents *coadaptation* between units: each unit is forced to learn to be useful by itself, because its utility cannot depend on the presence of other units in the same layer (as they can be removed at random).

1.3.4 Convolutional neural networks

Convolutional neural networks or *convnets* [43, 83] are a subclass of neural networks with constrained connectivity patterns between some of the layers. They can be used when the input data exhibits some kind of topological structure, like the ordering of image pixels in a grid, or the temporal structure of an audio signal.

Convolutional neural networks contain two types of layers with restricted connectivity: *convolutional layers* and *pooling layers*. We will describe the case of an input with a two-dimensional structure (e.g. an image) in some detail. A convolutional layer takes a stack of *feature maps* (e.g. the colour channels of an image) as input and convolves each of these with a set of learnable filters to produce a stack of output feature maps. This is ef-

ficiently implemented by replacing the matrix-vector product $\mathbf{W}_n\mathbf{x}_{n-1}$ in Equation 1.1 with a sum of convolutions. We represent the input of layer n as a set of K matrices $\mathbf{X}_{n-1}^{(k)}$, with $k = 1 \dots K$. Each of these matrices represents a different input feature map[9]. The output feature maps $\mathbf{X}_n^{(l)}$, $l = 1 \dots L$ are represented as follows:

$$\mathbf{X}_n^{(l)} = f\left(\sum_{k=1}^{K} \mathbf{W}_n^{(k,l)} * \mathbf{X}_{n-1}^{(k)} + b_n^{(l)}\right). \tag{1.4}$$

Here, $*$ represents the two-dimensional convolution operation, the matrices $\mathbf{W}_n^{(k,l)}$ represent the *filters* of layer n, and $b_n^{(l)}$ represents the bias for feature map l. Note that a feature map $\mathbf{X}_n^{(l)}$ is obtained by computing a sum of K convolutions with the feature maps of the previous layer. The bias $b_n^{(l)}$ can optionally be replaced by a matrix $\mathbf{B}_n^{(l)}$, so that each spatial position in the feature map has its own bias ('untied' biases). This allows the sensitivity of the filters to vary across the input.

By replacing the matrix product with a sum of convolutions, the connectivity of the layer is effectively restricted to take advantage of the input structure and to reduce the number of parameters. Each unit is only connected to a local subset of the units in the layer below, and each unit is replicated across the entire input. This is shown in the left side of Figure 1.4. This means that each unit can be seen as detecting a particular feature across the input (for example, an oriented edge in an image). Applying feature detectors across the entire input enables the exploitation of translational symmetry in images.

As a consequence of this restricted connectivity pattern, convolutional layers typically have far fewer parameters than traditional *dense* (or *fully-connected*) layers that compute a transformation of their input according to Equation 1.1. This reduction in parameters can drastically improve generalization performance and make the model scale to larger input dimensionalities.

Because convolutional layers are only able to model local correlations in the input, the dimensionality of the feature maps is often reduced between convolutional layers by inserting pooling layers. This allows higher layers to model correlations across a larger part of the input, albeit with a lower resolution. A pooling layer reduces the dimensionality of a feature map by computing some aggregation function (typically the maximum or the mean) across small local regions of the input [20], as shown in the right side of Figure 1.4. This also makes the model invariant to small translations of the

[9]Note that this depends on the dimensionality of the data. In the case of a two-dimensional convolution, a set of matrices is appropriate. It could also be a set of vectors (1D) or tensors (3D).

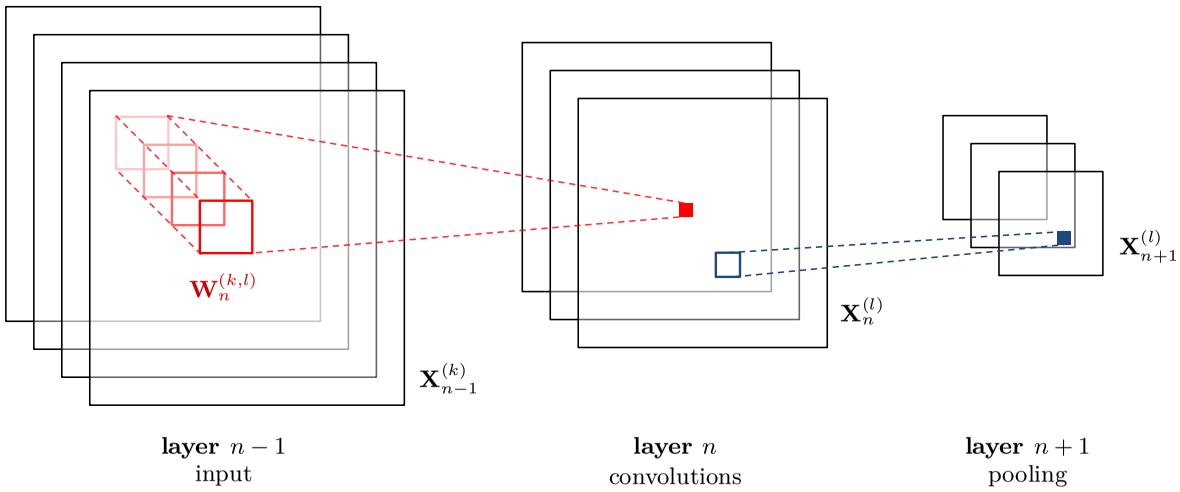


Figure 1.4: A schematic overview of a convolutional layer followed by a pooling layer: each unit in the convolutional layer is connected to a local neighborhood in all feature maps of the previous layer. The pooling layer aggregates groups of neighboring units from the layer below.

input, which is a desirable property for modelling images and many other types of data. Unlike convolutional layers, pooling layers typically do not have any trainable parameters.

By alternating convolutional and pooling layers, higher layers in the network see a progressively more coarse representation of the input. As a result, these layers are able to model higher-level abstractions more easily because each unit is able to see a larger part of the input.

Convolutional neural networks constitute the state of the art in many computer vision problems. Since their effectiveness for large-scale image classification was demonstrated, they have been ubiquitous in computer vision research [73, 118, 145, 132].

1.3.5 Deep learning for MIR

As mentioned before, music is hierarchically structured in many different ways. In the context of MIR, this structure can be exploited by building hierarchical models. Deep neural networks, with their layers that build upon representations extracted by other layers further down in the hierarchy, are able to do this efficiently while requiring minimal engineering effort: the learning procedure is able to map the different layers of a model to different levels of the musical hierarchy autonomously. This automatic mapping of the hierarchical structure of the model to the hierarchy of the task at hand has been observed before in e.g. computer vision tasks.

There is a very strong tradition of *feature engineering* in MIR: feature representations for music are constructed based on detailed analysis and intricate knowledge of how musical audio signals are structured. Another common approach is to reappropriate features designed for related problems such as speech recognition. Both of these approaches have proven effective, but are showing signs of stagnation [67]. The ability of deep learning models to discover good representations autonomously largely obviates the need for handcrafting feature representations. In both computer vision and speech recognition, feature learning has all but displaced approaches based on feature engineering, because it attains better performance at a reduced engineering cost and requires less domain knowledge. The same trend is now starting to materialize in MIR.

1.4 Research contributions

Compared to more established fields like speech recognition and computer vision, where approaches based on deep learning are well-established, rel-

atively few researchers are working on MIR problems and an even smaller fraction of those are exploring deep learning for this purpose. The work described in this thesis represents an exploration of deep learning and feature learning for audio-based music classification, tagging and recommendation. This section provides an overview of the research contributions and an outline of the remainder of this thesis.

An appendix describes work on image classification using deep neural networks that was done in the context of an international data science competition. This work falls outside the main theme of this dissertation but nevertheless constitutes a significant part of my work as a PhD student.

Deep learning is a burgeoning and rapidly evolving field of research: new methods were being introduced and swiftly adopted during my time as a PhD student, and the state of the art for a variety of problems has advanced quickly. The same is still true today. Some of the work described in this thesis was done several years ago, when different approaches were popular and the available hardware for building and training deep learning models was much less powerful than it is today. For this reason, each of the following chapters is introduced by a brief discussion of the context in which the research occurred. I will also pay special attention to recent developments in the conclusion of each chapter, and discuss how the results should be interpreted and what their relevance is today.

Music classification with a pre-trained convnet (Chapter 2)

The *Million Song Dataset* contains audio features and metadata for one million songs. In this chapter, we build a convolutional network that is trained on this dataset to perform artist recognition, genre recognition and key detection. The network is tailored to summarize the audio features over musically significant timescales. Because labels are scarce, it is infeasible to train the network on all available data in a supervised fashion. We use unsupervised pre-training to be able to harness the entire dataset: we train a convolutional deep belief network on all data, and then use the learnt parameters to initialize a convolutional multilayer perceptron with the same architecture. The MLP is then finetuned on a labeled subset of the data for each task.

Multiscale feature learning for music audio (Chapter 3)

Content-based music information retrieval tasks are typically solved with a two-stage approach: features are extracted from music audio signals, and are then used as input to a regressor or classifier. These features can be

engineered or learnt from data. Although the former approach was dominant in the past, feature learning has started to receive more attention from the MIR community in recent years. There has also been increased interest in multiscale representations of music audio recently. Such representations are more versatile because music audio exhibits structure on multiple timescales, which are relevant for different MIR tasks to varying degrees.

Inspired by recent results in feature learning using very simple and fast algorithms such as K-means to great effect, we propose three different architectures for multiscale audio feature learning using the spherical K-means algorithm. We evaluate them for automatic tagging, similarity metric learning and genre recognition on the Magnatagatune and GTZAN datasets, as well as subsets of the Million Song Dataset.

Deep content-based music recommendation (Chapter 4)

Automatic music recommendation has become an increasingly relevant problem in recent years, since a lot of music is now sold and consumed digitally. Most recommender systems rely on collaborative filtering. However, this approach suffers from the cold start problem: it fails when no usage data is available, so it is not effective for recommending new and unpopular songs.

In this chapter, we propose to use a latent factor model for recommendation, and predict the latent factors from music audio when they cannot be obtained from usage data. We compare a traditional approach using a bag-of-words representation of the audio signals with deep convolutional neural networks, and evaluate the predictions quantitatively and qualitatively on the Million Song Dataset. We show that using predicted latent factors produces sensible recommendations, despite the fact that there is a large semantic gap between the characteristics of a song that affect user preference and the corresponding audio signal. We also show that recent advances in deep learning translate very well to the music recommendation setting, with deep convolutional neural networks significantly outperforming the traditional approach.

This work was done in close collaboration with my colleague Aäron van den Oord.

End-to-end learning for music audio (Chapter 5)

Content-based music information retrieval tasks have traditionally been solved using engineered features and shallow processing architectures. In recent years, there has been increasing interest in using feature learning and deep architectures instead, thus reducing the required engineering effort and the

need for prior knowledge. However, this new approach typically still relies on mid-level representations of music audio, e.g. spectrograms, instead of raw audio signals.

In this chapter, we investigate whether it is possible to apply feature learning directly to raw audio signals. We train convolutional neural networks using both approaches and compare their performance on an automatic tagging task. Although they do not outperform a spectrogram-based approach, the networks are able to autonomously discover frequency decompositions from raw audio, as well as phase- and translation-invariant feature representations.

Conclusions and perspectives (Chapter 6)

Finally, overarching conclusions and some perspectives for future work are given in this chapter.

Galaxy morphology classification with convnets (Appendix A)

Measuring the morphological parameters of galaxies is a key requirement for studying their formation and evolution. Morphological analysis has traditionally been carried out mostly via visual inspection by trained experts, which is time-consuming and does not scale to large numbers of images.

Although attempts have been made to build automated classification systems, these have not been able to achieve the desired level of accuracy. The Galaxy Zoo project successfully applied a crowdsourcing strategy, inviting online users to classify images by answering a series of questions. Unfortunately, even this approach does not scale well enough to keep up with the increasing availability of galaxy images.

In this appendix, we present a deep neural network model for galaxy morphology classification which exploits translational and rotational symmetry. It was developed in the context of the *Galaxy Challenge*, an international competition to build the best model for morphology classification based on annotated images from the Galaxy Zoo project.

1.5 List of publications

Journal publications

1. **Dieleman S.**, Willett K., Dambre J. (2015). Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Montly*

Notices of the Royal Astronomical Society, 450(2).

2. Verstraeten D., Schrauwen B., **Dieleman S.**, Brakel P, Buteneers P., Pecevski D. (2012). Oger: modular learning architectures for large-scale sequential processing. *Journal of Machine Learning Research*, 13.

Conference publications

1. van den Oord A., **Dieleman S.**, Schrauwen B. (2014). Transfer learning by supervised pre-training for audio-based music classification. *Proceedings of the 15th international society for music information retrieval conference (ISMIR 2014).*

2. **Dieleman S.**, Schrauwen B. (2014). End-to-end learning for music audio. *International Conference on Acoustics Speech and Signal Processing (ICASSP 2014).*

3. Pigou L., **Dieleman S.**, Kindermans P.J., Schrauwen B. (2014). Sign language recognition using convolutional neural networks. *European Conference on Computer Vision, Workshop.*

4. Caluwaerts K., wyffels F., **Dieleman S.**, Schrauwen B. (2013). The spectral radius remains a valid indicator of the echo state property for large reservoirs *IEEE International Joint Conference on Neural Networks (IJCNN).*

5. **Dieleman S.**, Schrauwen B. (2013). Multiscale approaches to music audio feature learning. *Proceedings of the 14th international society for music information retrieval conference (ISMIR 2013).*

6. van den Oord A., **Dieleman S.**, Schrauwen B. (2013) Deep content-based music recommendation. *Advances in Neural Information Processing Systems 26.*

7. van den Oord A. **Dieleman S.**, Schrauwen B. (2013) Learning a piecewise linear transform coding scheme for images. *Proceedings of SPIE, the International Society for Optical Engineering.*

8. Brakel P., **Dieleman S.**, Schrauwen B. (2012) Training restricted Boltzmann machines with multi-tempering: harnessing parallelization. *Lecture notes in computer science.*

9. **Dieleman S.**, Schrauwen B. (2012). Accelerating sparse restricted Boltzmann machine training using non-Gaussianity measures. *NIPS workshop: Deep learning and unsupervised feature learning.*

10. **Dieleman S.**, van den Oord A., Schrauwen B. (2012). Parallel one-versus-rest SVM training on the GPU. *NIPS workshop: Big learning: algorithms, systems and tools.*

11. **Dieleman S.**, Brakel P., Schrauwen B. (2011). Audio-based music classification with a pretrained convolutional network. *Proceedings of the 12th international society for music information retrieval conference (ISMIR 2011).*

Abstracts, demonstrations, presentations

1. **Dieleman S.** (2015). Deep content-based music recommendation. *Machine Learning for Music Discovery Workshop at the International Conference on Machine Learning (ICML 2015)*, invited talk.

2. van den Oord A., **Dieleman S.**, Schrauwen B. (2013). Deep content-based music recommendation. *Neural Information Processing Systems 26*, demonstration.

3. **Dieleman S.**, Schrauwen B. (2012). Learning content-based metrics for music similarity. *5th International Workshop on Machine Learning and Music*, abstract.

Software

1. **Dieleman S.**, Schlüter J., Raffel C., Olson E., Kaae Sønderby S., Nouri D., van den Oord A., Battenberg E. (2015). Lasagne: first release.

2

Music classification with a pre-trained convolutional neural network

2.1 Introduction

In 2011, the Laboratory for the Recognition and Organization of Speech and Audio (LabROSA)[1] of Columbia University released a large music dataset consisting of audio features and metadata for one million songs, aptly named the 'Million Song Dataset' (MSD) [18]. One of the purposes of the dataset was to encourage research on algorithms that scale to commercial datasets, which are typically many orders of magnitude larger than those used in academia, especially in the domain of music information retrieval.

Around the same time, some researchers started to look into applications of deep learning to MIR problems. Deep learning methods scale very well with dataset size, and many such methods arguably only work well when the dataset on which they are trained is sufficiently large. The release of the MSD came at the right time to be used for this kind of research. I have made extensive use of the dataset throughout my PhD research.

Additionally, the MSD is ideally suited for experiments involving unsupervised pre-training: most of the provided types of metadata are only available for a subset of the dataset, so for many tasks it is not possible to train models in an end-to-end supervised fashion on the entire dataset. Instead, subsets of the dataset with strong, reliable labels for a specific task can be constructed, and the remainder of the dataset (often an order of magnitude larger than the selected subset) can be used for unsupervised pre-training. This is the approach I took for this work.

In this chapter, a model for classifying songs according to their genre, artist and key is developed: a convolutional network that summarizes input

[1] http://labrosa.ee.columbia.edu/

feature representations of songs over musically significant timescales.

Developing techniques that can harness the entire dataset is quite a challenge. I use the majority of the data in an unsupervised learning phase, where the network learns to model the audio features. Due to its size, the dataset is very suitable for unsupervised learning. This is followed by a supervised training phase, where only a small task-specific subset of the dataset is used to train a discriminative model using the same network. I have investigated the gains that can be achieved by using a convolutional architecture, and the additional gains that unsupervised pre-training can offer.

This chapter is structured as follows: the layout of the dataset is detailed in Section 2.2. An overview of convolutional neural networks and layerwise pre-training follows in Section 2.3. Section 2.4 describes the classification tasks that were used to evaluate the model. Section 2.5 provides an overview of our approach, and Section 2.6 describes our experimental setup. Results are given in Section 2.7.

2.2 Dataset

2.2.1 The Million Song Dataset

The Million Song Dataset is a collection of all the information that is available through The Echo Nest API[2] for one million popular songs. This means that a lot of the data was automatically derived from musical audio signals, which should be taken into account when it is used for learning. Available metadata includes artist and album information and the year of the performance. Musical information derived directly from the audio signal includes the key, the mode and the time signature. User-assigned tags are also available.

The audio features in the dataset were obtained by first dividing each song into so-called segments. Segment boundaries roughly correspond to onsets of notes or other musical events. For each segment, a feature vector consisting of 12 timbre and 12 chroma components was computed, as well as the maximal loudness within the segment.

The chroma features describe the pitch content of the music. Each of the 12 components corresponds to a pitch class (ranging from C to B). Their values indicate the relative presence of the pitches, with the most prominent one always having a value of 1. All components lie within the interval $[0, 1]$. The timbre features are the coefficients of 12 basis functions which capture certain timbral characteristics like brightness, flatness and

[2] http://the.echonest.com/

attack. They are unbounded and roughly centered around 0. More details about this representation can be found in the documentation for The Echo Nest's *analyzer* tool [70].

Unfortunately, the automated methods used to build the dataset lead to the presence of a relatively large number of duplicate tracks. When the dataset is divided into a train and a test set in a naive fashion, some examples might occur in both subsets, which is undesirable. Luckily, the authors of the dataset have published an extensive list of known duplicates[3]. Using this list, over 78,000 tracks were removed.

2.2.2 Beat-aligned features

Although the segmentation that was performed to compute the audio features has its merits, we are more interested in *beat-aligned* features such as those used in [17]. The beat is the basic unit of time in music. Chord progressions and changes in musical texture tend to occur on the beat, and seeing as it is one of our goals to encode these characteristics in higher level features, it makes sense to use beat-aligned features as a starting point.

The features from the dataset can be converted to beat-aligned features using the rhythm information that is also supplied. The segments are mapped to beats, and then the feature vectors for all segments corresponding to the same beat are averaged.

2.3 Background

Deep learning is a fairly recent trend in machine learning: traditionally, the use of deep architectures with many layers of processing [8] was not very popular because they were very difficult to train. In 2006, Hinton demonstrated a fast training method for *deep belief networks* (DBNs), a particular type of deep model [61]. This led to a surge in popularity of these models, establishing *deep learning* as a new area of research. Deep belief networks are probabilistic generative models, which are obtained by stacking multiple restricted Boltzmann machines (RBMs) on top of each other.

2.3.1 Restricted Boltzmann machines

A restricted Boltzmann machine is a probabilistic model consisting of a set of visible units and a set of hidden units which form a bipartite graph; there

[3] http://labrosa.ee.columbia.edu/millionsong/sites/default/files/AdditionalFiles/msd_duplicates.txt

are no connections between pairs of visible units or pairs of hidden units, but every visible unit is connected to every hidden unit. They are a kind of undirected graphical model. A schematic representation is shown in Figure 2.1.

The visible units of an RBM correspond to the input variables of the data that is to be modelled. In image processing, each visible unit typically represents one pixel. The hidden units capture correlations between visible units and can be seen as *feature detectors*. The model learns the underlying distribution of the data by representing it in terms of features that are derived from the data itself.

Each connection has a particular weight, and each of the units can also have a bias. These trainable parameters can be learnt from data. Unfortunately, maximum likelihood learning is intractable in RBMs. Instead, the *contrastive divergence* learning rule, which is an approximation to maximum likelihood learning, can be used [59].

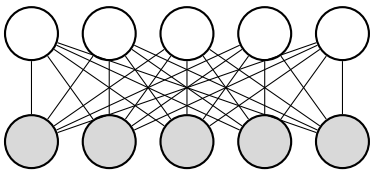


Figure 2.1: Schematic representation of an RBM, with the visible units at the bottom and the hidden units at the top. Note how there are no lateral connections between two visible or two hidden units.

RBMs typically consist of binary units, which can be on or off. This makes sense for the hidden units, which are feature detectors, but it is not always the best choice for the visible units. It is also possible to construct an RBM for continuous data, with Gaussian visible units[4].

2.3.2 Deep belief networks

A deep belief network (DBN) can be represented as a hybrid graphical model with both directed and undirected connections. It consists of multiple RBMs stacked on top of each other, with the hidden units of RBM i being used as visible units of RBM $i+1$. The bottom RBM learns a shallow model of the data. The next one then learns to model the hidden units of the first, and so on: higher-level features are extracted from lower-level features. Each RBM is trained separately. The parameters of all these RBMs are then used to

[4]In fact, any distribution from the exponential family can be used, but binary (Bernoulli) and Gaussian units are the most commonly used types.

construct the DBN, which has undirected connections between the top two layers of units (as in an RBM) and directed connections elsewhere.

Top-level features learnt by DBNs can be used to train discriminative models. In this fashion, they have been applied succesfully to image processing problems like handwriting recognition [61] and object recognition [74], but also to classification of audio signals [85], and even music classification [51]. For a detailed technical overview of RBMs and DBNs, see [8].

2.3.3 Convolutional deep belief networks

Convolutional neural networks have previously been discussed in Section 1.3.4. Restricted Boltzmann machines and deep belief networks can also be made convolutional by restricting the connectivity between their units, and by inserting (max-)pooling operations between layers. Figure 2.2 shows a stack consisting of a convolutional RBM and a max-pooling layer. This combination can again be stacked multiple times to create a convolutional deep belief network (CDBN). Convolutional deep belief networks have been used for object recognition [113, 84], and to extract features from audio signals, for speech recognition as well as for music classification [85].

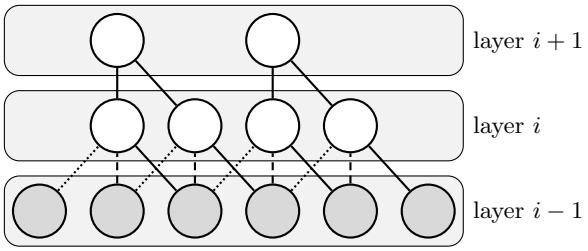


Figure 2.2: A max-pooling layer ($i+1$) stacked on top of a convolutional layer (i). Note that layer $i-1$ and layer i are not fully connected. The connections are drawn in different styles to indicate which weights are shared.

Convolutional models are typically used for image processing, where stronger correlations between nearby pixels and the translation invariance of image features are exploited to significantly reduce the number of parameters. Audio signals have similar characteristics, although the locality is temporal rather than spatial. Whereas convolutions are typically applied in two dimensions in the context of images, those applied to audio data are often one-dimensional instead.

It is possible to use 2D convolutions when the input consists of time-frequency representations of audio signals. In that case, convolution along

the frequency axis in combination with pooling allows for transposition invariance. We briefly explore this in Section 3.5.5.

2.3.4 Supervised finetuning

As mentioned earlier, top-level DBN features can be used as input for a classification method; common choices are support vector machines or logistic regression. We can train a logistic regression classifier by gradient descent, using the DBN model to extract feature representations from the input data.

It is also possible to convert a DBN into a multilayer perceptron (MLP). We can simply reuse the weights of the interconnections and the biases of the hidden units. We then stack a logistic regression layer on top of this MLP and train the whole model jointly using gradient descent. This approach is called *supervised finetuning*: the DBN weights that were initially learnt to model the data are now finetuned for a specific discriminative task using backpropagation.

2.4 Tasks

We performed several classification tasks on music tracks: artist recognition, genre recognition and key detection. Labeled datasets for each of the tasks were extracted from the Million Song Dataset. Three (partially overlapping) subsets were selected:

- artist recognition: the 50 artists with the most tracks in the dataset were identified, and 100 tracks of each artist were selected (5000 tracks in total);
- genre recognition: 20 common genres were selected manually using tags[5] that are included in the dataset: folk, punk, metal, jazz, country, blues, classical, rnb, new wave, world, soul, latin, dance, reggae, techno, funk, rap, hip hop, rock and pop. For each genre, 250 tracks were selected (5000 tracks in total);
- key detection: the key information in the dataset was automatically annotated, so it may be unreliable. To avoid problems with incorrect labels, we selected 250 tracks with a high key confidence for each of the 12 possible keys (3000 tracks in total). Nevertheless, the results for this task should be taken with a grain of salt, as what is actually being measured is how well the network can reproduce the results of the algorithm that was used to compute the annotations. Note that we also assume that each track is

[5] The dataset provides different kinds of tags. We used the MusicBrainz tags because these are the most reliable [18].

associated with a single key (i.e. there is no modulation between multiple keys). This is not necessarily true in practice.

The subsets were then divided into balanced train, validation and test sets according to a 80% / 10% / 10% split.

2.5 Approach

We built a convolutional network, designed to aggregate the features from the dataset on musically significant timescales. Properties that are typical for certain genres, artists or keys should become apparent at this level. We used the same network to tackle all three classification tasks, demonstrating the versatility of the learned features.

To train the network, we first performed layerwise unsupervised pre-training using an RBM for each layer on the entire Million Song Dataset[6]. We then trained and evaluated the network as an MLP with backpropagation for each of the classification tasks. We used the Theano Python library to implement all experiments, so they could be GPU-accelerated easily [14].

2.5.1 Network layout

The input of the network consists of beat-aligned chroma and timbre features for a given track, so there are 24 input dimensions in total. The maximal loudness component was not used, as the timbre features already include a loudness component. Note that tracks vary considerably in length, but the convolutional nature of the network allows us to cope easily with variable-length input.

First, we separated the chroma and timbre features into two input layers (layers 0a and 0b). Then, separate convolutional layers were stacked onto both input layers (layers 1a and 1b). These layers learn features with a width of 8 beats. It was observed that most of the tracks in the dataset have a 4/4 time signature (which is also true for contemporary music in general). This means that there are 4 beats in a bar. The width of the features was chosen to be two bars, seeing as this is the timescale on which chord progressions and changes in musical texture are most likely to occur. We used 100 feature maps for each layer.

By using two separate layers, the network does not learn correlations between chroma and timbre features at this level. This allows it to focus on learning correlations between timbre components and between chroma

[6] Excluding known duplicates and tracks used for validation and testing for any of the tasks.

components separately; such correlations are likely to be easier to discover. A similar approach was used in [112] to learn features over multiple data modalities.

The output of the convolutional layers was then max-pooled in the time dimension with a pool size of 4 (layers 2a and 2b). Once again, we made use of the observation that most of the tracks in the dataset have a 4/4 time signature, with 4 beats per bar; the output of the max-pooling layer is invariant to all displacements of less than one bar (up to 3 beats).

The max-pooled outputs of both layers were then concatenated, yielding 200 features with a granularity of approximately 1 bar. We stacked another convolutional layer with 100 feature maps on top of this, which learns features with a width of 8 bars (layer 3). This width was selected because musical themes are often contained within a length of 8 bars. Correlations between timbre and chroma components can now be discovered as well.

Finally, another max-pooling layer with a pool size of 4 was added (layer 4). The features obtained from this layer have a granularity of 4 bars and a scope of roughly 8 bars. To perform the classification tasks, a fifth layer performing logistic regression was added. Classification is performed separately for each timestep on layer 4, and the resulting posterior distributions are averaged to obtain a posterior distribution for an entire track. The most probable class is then selected. The layout of the network is shown in Figure 2.3.

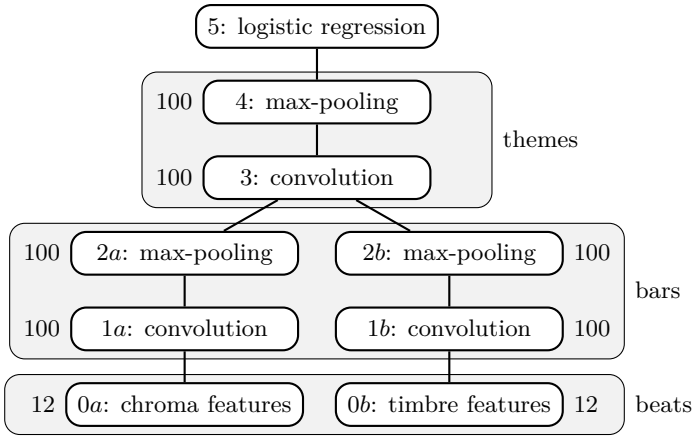


Figure 2.3: The network layout. The number of dimensions or feature maps for each layer is indicated on the side. The layers have also been grouped according to the timescale on which they operate.

2.5.2 Unsupervised pre-training

It would be impossible to train the network in a supervised fashion with the entire Million Song Dataset. This is this computationally infeasible, and on top of that the provided labels are not perfect; some are missing, others are incorrect or have a very low confidence.

As mentioned before, I pre-trained the network using timbre and chroma features for all tracks in the dataset. I used the beat-aligned chroma features directly as inputs to the network; the timbre features were first normalized per track to have zero mean and unit variance.

To train the RBM in layer 1b (timbre), I used Gaussian visible units, which allow for continuous input data to be modeled. For layer 1a (chroma), I used binary units. Technically, this is not possible because the chroma features are continuous values that lie between 0 and 1, whereas binary units are intended to model binary variables. However, we can interpret these values as probabilities and sample from them, yielding binary input data. In practice, we do not perform this sampling explicitly, but we use the *mean field approximation* (see Section 2.5.2.2). In RBMs, learning is much more stable for binary units than for Gaussian units, so being able to use binary units is a significant advantage.

I used single step constrastive divergence (CD-1) everywhere. A learning rate of 0.005 was used to train the RBMs with binary visible units; a learning rate of 0.0001 was used for the RBM with Gaussian visible units. We performed only a single run through the entire dataset. Due to its size, performing multiple epochs turned out to be unnecessary (and would require too much computation time).

2.5.2.1 Sparsity

I modified the hidden unit activations according to [46] to encourage them to be sparse. Convolutional RBMs are overcomplete models, so adding a sparsity penalty term ensures that the learnt feature representations are useful [85]. In addition, sparse activations are essential for max-pooling to work properly [20, 124].

I used a target activation probability of 0.05 for layers 1a and 1b, and a target of 0.1 for layer 3. A relative sparsity cost of 0.1 was used in all cases. Please refer to Hinton [60] for more information about these parameters.

2.5.2.2 Mean field approximation

Where possible, I eliminated sampling steps in the RBM training process by using the mean field approximation: instead of sampling from a distri-

bution, its mean value is used instead. This eliminates sampling noise and often positively affects convergence. I used this for the chroma inputs and in the contrastive divergence algorithm, except when updating the hidden states, as recommended in [60]. Interpreting continuous input values that are constrained to a finite interval as input probabilities to train an RBM is common practice [59].

2.6 Experiments

I trained the network as a convolutional MLP for each of the classification tasks described in Section 2.4: first with random initialization of the weights, and then using the weights learnt by unsupervised pre-training (supervised finetuning), yielding six different experiments. I tried learning rates of 0.05, 0.005 and 0.0005 and trained for 30 epochs. To initialize the random weights, I sampled them from a Gaussian distribution with a mean and variance corresponding to those of the weights learnt by the DBN. This ensures that the results are comparable.

I also trained a naive Bayes classifier and a logistic regession classifier that operate on time windows of features from the dataset, resulting in six more experiments. I chose a window size of 32 beats (8 bars), which is comparable to the timescale on which the convolutional network operates. For the logistic regression classifier, I tried learning rates of 0.005, 0.0005, $5 \cdot 10^{-5}$, $5 \cdot 10^{-6}$ and $5 \cdot 10^{-7}$ and also trained for 30 epochs. Both the chroma features and the timbre features were normalized to have a zero mean and a unit variance in this case.

For each of the twelve experiments, we determined the optimal parameters using the validation sets, and then computed the classification accuracies on the test sets using these parameters. The results can be found in Table 2.1.

2.7 Results

The first thing to notice is that the key detection task seems to be fairly simple. The achieved accuracies are much higher than for the other tasks, and the simplest technique performs best. There are multiple possible explanations for this:

- the property we are trying to determine is quite low-level. The key of a track is in a very close relationship with the chroma features and

		genre recognition	artist recognition	key detection
	naive Bayes	10.02%	6.80%	73.74%
30 epochs	windowed logistic regression	25.90% ($5 \cdot 10^{-6}$)	32.13% ($5 \cdot 10^{-5}$)	**86.53%** ($5 \cdot 10^{-5}$)
	conv. MLP without pre-training	**29.52%** (0.005)	34.34% (0.05)	83.84% (0.05)
	conv. MLP with pre-training	29.12% (0.005)	**35.74%** (0.05)	83.84% (0.005)
20 epochs	conv. MLP without pre-training	24.90% (0.05)	33.94% (0.05)	83.84% (0.05)
	conv. MLP with pre-training	**27.31%** (0.005)	**35.54%** (0.05)	**84.51%** (0.005)

Table 2.1: Test accuracies and corresponding learning rates for each of the classification tasks, with and without pre-training.

how they evolve through time. Relating the genre or the artist to these features is much more difficult;

- to construct the dataset for this task, I selected tracks with a high key confidence. This implies that the algorithm used to annotate key information in the Million Song Dataset could identify the key of these tracks with relative ease. It would make sense that the same is true for our models. Unfortunately, there is no way to verify this, except by constructing a manually labeled dataset.

For the other tasks, the convolutional network has a definite edge over the other approaches: the classification accuracies increase significantly.

The gains obtained with pre-training on the other hand seem to be much more modest; this is only advantageous for the artist recognition task, which is quite difficult because it is a 50-way classification problem. The utility of pre-training for this task could stem from the fact that the number of tracks per class available for training (80) is much lower compared to the other tasks (200). Indeed, it has been shown that gains from unsupervised pre-training are maximal when the amount of available labeled training data is limited [35]. It is also worth noting that this data scarcity is intrinsic to the task at hand, and not just an artefact of the way the dataset was constructed - few artists have a discography with more than 100 tracks.

The optimal learning rate for key detection with the convolutional network differs depending on whether pre-training is used or not. This is because the training for this task without pre-training did not converge after 30 epochs using a learning rate of 0.005. This indicates that convergence is faster when pre-training is used. To investigate this, we also compared classification accuracies obtained after only 20 training epochs, which can be found in the bottom half of Table 2.1. We now observe that pre-training is beneficial for all tasks. This confirms our intuition that it improves convergence speed.

2.8 Conclusion

I have trained a convolutional network on beat-aligned timbre and chroma features obtained from music audio data to perform a number of classification tasks. The convolutional nature of the network allowed it to summarize these features over musically significant timescales, leading to an increase in accuracy. I used unsupervised pre-training with a very large dataset, which improved convergence speed and, for the artist recognition task, classification accuracy. It is clear that the ability to harness a large amount of unlabeled

data is advantageous for tasks where the amount of available training data is limited.

The work described in this chapter was done in the first half of 2011 and predates some high-impact innovations and profound changes in the way convolutional neural networks are typically built and trained. This includes the use of rectified linear units (ReLUs) [110, 45] and dropout regularisation [138], among other things. Unsupervised pre-training has also lost a lot of its importance since then, but is still a viable approach in this problem setting today because of the abundance of data and the relative lack of useful labels.

The trained classifiers achieve relatively poor results overall. This is partly because the genre and artist classification tasks are quite challenging, but also due to how the datasets were constructed. The labels for the genre classification set come from tags, which may not always be reliable and they may potentially overlap. But most importantly, the input representation, i.e. beat-aligned precomputed features, is far from optimal. Unfortunately no other input representation was available at the time because I did not have access to the raw audio data, which was not included in the dataset. This is unfortunate because deep neural networks are especially suitable for learning task-specific features from raw data. Having to learn high-level features from a small set of pre-computed low-level features severely reduces their effectiveness.

For further work on this dataset, described in Chapters 3 and 4, my colleague Aäron van den Oord and I managed to obtain 30 second clips for more than 99% of the tracks in the MSD. This data would also have been very useful for the work described in this chapter, although presumably learning features from audio signals or a time-frequency representation extracted from the audio signals would have required a deeper network. Neural network models with more than 4 or 5 layers had not been explored much at that point in time, and the computational requirements for training are proportional to the number of layers, so this may not have been feasible either. Their training was also much more difficult due to the use of sigmoidal nonlinearities. Modern neural networks typically use piecewise linear activation functions and sometimes have more than 20 layers. Such 'very deep' models should be capable to achieve much better results on these tasks using audio signals or mid-level representations as input.

Despite these shortcomings, this work shows the value of unlabeled data and unsupervised pre-training to harness this data. This observation is still relevant today because for many problems that can be tackled using a deep learning approach, both inside and outside MIR, unlabeled data is abundant whereas labels tend to be scarce and expensive to obtain.

Although the first applications of deep learning methods to MIR problems predate this work (most notably the work of Hamel et al. [52, 51] and

Lee et al. [84]), the availability of the MSD made it possible to explore unsupervised pre-training in the context of MIR on a much larger scale. Since then, feature learning for music has only gained popularity, with some researchers arguing for a paradigm shift in the MIR community and moving away from the deep-rooted tradition of manual feature design [67].

Feature learning for music using convolutional neural networks is also explored in chapters 4 and 5, but the learning procedure is fully supervised in both cases. An alternative unsupervised approach based on the K-means algorithm is described in chapter 3.

3 Unsupervised multiscale feature learning

3.1 Introduction

Feature learning and deep learning are often conflated, but not all feature learning is necessarily deep: it is possible to learn shallow (i.e. single-layer) representations of data, and this can also be useful. It often has significant advantages in terms of computation cost as well.

In the previous chapter, I used a neural network to learn a hierarchy of features. Here, I will explore a different approach to building a feature hierarchy, using shallow feature learning with the spherical K-means algorithm on multiple timescale representations of audio signals. In the resulting feature hierarchy, higher-level features do not build upon lower-level features, but extract patterns at coarser timescales instead. In other words, the hierarchy is entirely temporal: the features do not differ in terms of their level of abstraction, but only in the timescale of the patterns they detect.

This alternative approach has some advantages; for example, the learning step is usually quick and can be executed in parallel for all levels of the hierarchy. The learnt features are also easy to interpret due to their low level of abstraction. Finally, the K-means algorithm is familiar to many researchers in the MIR domain, and its spherical variant is easy to understand and implement.

The rest of this chapter is structured as follows: the use of features in MIR is briefly summarised in Section 3.2, and feature learning is discussed in Section 3.3. An overview of the multiple timescale representations considered in this chapter is given in Section 3.4. The proposed feature learning approach is described in Section 3.5. I demonstrate its versatility with a number of different tasks on three datasets. The experiments and results are given in Sections 3.6 and 3.7, and conclusions are drawn in Section 3.8.

A paper describing a preliminary version of this work was presented at the ISMIR 2013 conference [32]. This chapter describes the work in more detail and extends it in the following ways:

- the evaluation includes a wider range of different tasks, and two additional datasets;
- I have evaluated two extensions of the proposed approach: incorporating frequency invariance and learning a multi-layer hierarchy of features (see Sections 3.5.5 and 3.5.6 respectively).

3.2 Features for content-based MIR

A variety of different tasks and settings are studied in the context of content-based MIR, including genre classification, artist recognition, automatic tagging and music recommendation. Despite this, many content-based MIR systems have a very similar architecture, consisting of two stages: features are first extracted from music audio signals to transform them into a more meaningful representation. These features are then used as input to a regressor or a classifier, which is trained to perform the task at hand.

Casey et al. [23] and more recently Fu et al. [42] give an overview of the types of features that are commonly used for this purpose. All of them are the result of many years of research and engineering. They were developed using domain knowledge about the data and the specific problem that they were designed to solve. Nevertheless, they are often used for other types of problems, for which they may be suboptimal: for example, Mel-Frequency Cepstral Coefficients (MFCCs) were originally designed for speech recognition, but have been ubiquitous in MIR research ever since their introduction [40, 93].

Handcrafting features is costly and time-consuming, and it requires a lot of insight. In recent years, some researchers have applied *feature learning* techniques, which allow for feature representations to be learnt automatically from data, in an attempt to automate the search for good representations [85, 51, 56, 111]. I will discuss this approach in more detail in the next section.

Another recent development in MIR research is the increased interest in building systems that operate on *multiple timescales* [53, 3, 33, 41]. Music audio exhibits structure on many different timescales: at progressively longer timescales, we can identify notes, recurring motifs, phrases and themes. The various timescales are relevant for different tasks to varying degrees. I in-

vestigate a number of different multiple timescale representations of audio signals in Section 3.4.

3.3 Feature learning

3.3.1 Learning representations

The goal of feature learning is to learn transformations or representations of data that make the useful information contained within it more apparent [10]. For example, images are typically represented as grids of pixel intensity values. If one wants to recognise a certain object in an image, operating directly on such a representation makes this quite difficult. By transforming the image into a different representation first, the recognition task becomes much simpler. The same is true for musical audio signals, which are typically represented as discrete time series, describing how the signal amplitude varies over time. Such a representation is far removed from musical properties such as genre, for example, which is why the signal is usually first transformed into a different representation (such as MFCCs) that makes it easier to extract this type of information. However, as mentioned before, designing this transformation by hand is quite complex. Feature learning allows for this step to be automated.

A significant advantage of feature learning over manual feature design is that it frees up resources and time that can be spent on other research efforts instead. Feature learning algorithms also tend to be very reusable: they can be applied to different types of data and for different tasks, without requiring a lot of domain knowledge or manual finetuning. In many domains, including speech recognition and computer vision, feature learning approaches are now commonplace after improving on the previous state of the art in some tasks by a large margin [127, 82, 73].

In summary, rather than investing a considerable amount of time and effort in designing the best features for a specific task, using a feature learning approach instead is likely to result in similar or better performance at a lower engineering cost, and any insights and knowledge gained from this process are often applicable to other tasks as well.

3.3.2 Feature learning in MIR

In recent years, feature learning has started to receive more attention in the MIR community, with an increasing number of papers on the topic being presented at the ISMIR conference since 2010. Humphrey et al. [67] ar-

gue that the MIR community's reliance on handcrafted features combined with classifiers such as SVMs or logistic regression is hampering the field's progress. Instead they advocate the use of feature learning and deep processing architectures. They give a thorough overview of the advantages of this approach, and also describe some of its early applications in music informatics. Its adoption has been relatively limited so far, due to a number of reasons:

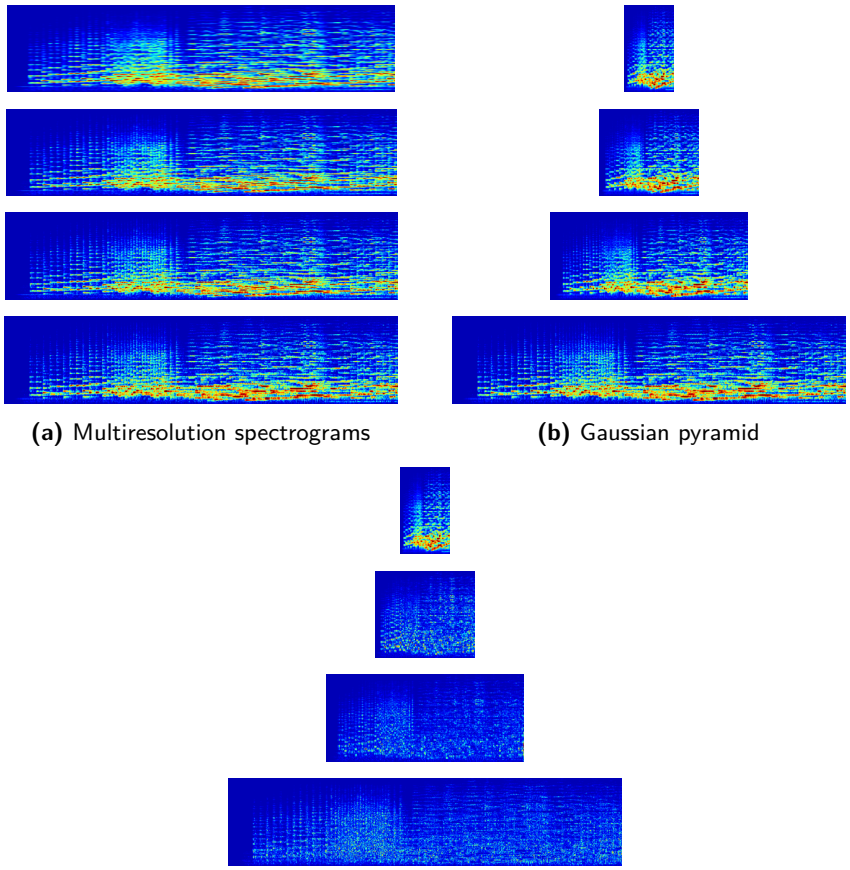
- Although it significantly reduces the amount of domain knowledge required to obtain a good representation, it requires knowledge about feature learning instead. To this day, no real off-the-shelf feature learning solutions are available that can be applied without understanding their internals.

- Many of the models that are currently used for feature learning require a large amount of computational power to train. Sometimes special hardware such as a GPU is necessary to be able to train them in a reasonable amount of time. These resources are not readily available to many researchers.

- Many feature learning techniques have a large number of hyperparameters that should be tuned, which is at best inconvenient, and at worst impossible to do without some knowledge about their significance.[1]

- Feature learning typically requires a large amount of training data. Unfortunately, data tends to be scarce in the context of music research, because copyright regulations and licensing issues prevent the distribution of large collections of music audio. In recent years, some datasets have been published that circumvent these issues [81, 18].

In image processing and computer vision, feature learning techniques are typically applied directly to raw pixel values. When working with audio signals however, the input is usually transformed into a time-frequency representation first, such as spectrograms (see Section 1.1.2). Time-frequency representations of audio signals are indispensible, because many higher-level characteristics of sound relate to energies in different frequency bands. Many techniques typically used for images can be applied to spectrograms as well, because they have the same two-dimensional shape.



(a) Multiresolution spectrograms

(b) Gaussian pyramid

(c) Laplacian pyramid

Figure 3.1: Three different 4-level multiscale time-frequency representations of music audio. Level 0 is at the bottom, higher levels correspond to coarser timescales.

3.4 Multiscale representations

For images and audio signals, feature learning algorithms are typically applied to small patches or fragments of the data. The underlying idea is that perceptual signals have a topological structure where neighbouring variables are strongly correlated, whereas variables that are farther away from each other are much less correlated. Furthermore, these local correlations tend to be similar everywhere. As a result, we can model the majority of the correlations by looking only at local neighbourhoods of variables, which considerably reduces the size of the models, and hence computational complexity.

Nevertheless, both images and audio signals tend to exhibit correlations at many different scales. Correlations at coarser scales are discarded by this approach, which is undesirable for many tasks. Especially in the case of music, a lot of structure is present at timescales much larger than the typical resolution of an audio signal. Multiscale representations alleviate this problem: they consist of multiple levels, each representing the input with a different resolution. We can then model local correlations at each level. At coarser scales, this means a larger part of the input will be taken into account.

In image processing, a popular method to obtain a multiscale representation of an image is to construct a *Gaussian* or *Laplacian pyramid* [22]. Each consecutive level of a Gaussian pyramid is obtained by smoothing and then subsampling the previous level by a factor of 2. The Laplacian pyramid can be derived from the Gaussian pyramid by computing the difference of each level and the level above it after upsampling. The top level remains the same. The result is that the representations at finer scales will not contain any information that is already represented at coarser scales. This reduces redundancy between the levels of the pyramid.

In the context of feature learning, Laplacian pyramids have been used as input to deep belief networks (DBNs) to learn features for image classification and phone recognition [146]. Farabet et al. [37] propose to use a multiscale convolutional network for scene labeling, using a Laplacian pyramid representation of images as its input, and achieves state of the art performance on standard scene parsing datasets.

In convolutional neural networks, subsampling layers are often inserted between the convolutional layers, so that features at each successive level take a larger part of the input into account and are increasingly translation invariant [83]. Although this results in a multiscale representation of the input, only the top level is connected to the output layer. The fine-grained

[1] However, recently Snoek et al. [137] and Bergstra et al. [13] have looked into automating this process as well, with promising results.

features are only used indirectly to construct the higher-level features. This approach has also been applied to music classification [88, 33] (see chapter 2).

3.4.1 Multiscale time-frequency representations of music audio

In the case of musical audio signals, multiple scales are typically only considered for the time axis, and not for the frequency axis, because it does not have the same topological structure. This is in contrast to images, where both axes are of the same nature. There is little to be gained by considering different levels of granularity in the frequency direction: to detect harmonic content, using the finest possible granularity will give the best results, and for inharmonic features, the precise granularity of the frequency axis does not matter much since they tend to span the entire frequency range anyway. In the time direction on the other hand, being able to detect features at multiple scales is much more useful. We will explore three approaches to obtain multiscale time-frequency representations of audio signals:

- **multiresolution spectrograms**: we can vary the size of the time windows used to compute spectrograms to obtain a multiscale representation. For each consecutive level, we will double the window size of the previous level. This approach was previously investigated by Eyben et al. [36] and Hamel et al. [54]. Although we could also double the hop size to reduce the size of the higher level representations, Hamel et al. found this to decrease performance considerably, so it is kept the same for all levels. Figure 3.1a shows an example of a set of multiresolution spectrograms. Note the decreased temporal resolution and increased frequency resolution in the higher levels.

- **Gaussian pyramid**: in this case, spectrograms are extracted only at the finest timescale and all higher levels can be obtained from them. Higher-level representations in a Gaussian pyramid will be smaller because of the subsampling step. An example of a Gaussian pyramid is shown in Figure 3.1b.

- **Laplacian pyramid**: as described earlier, this can be easily computed from the Gaussian pyramid representation. An example of a Laplacian pyramid is shown in Figure 3.1c. The top level of the pyramid is the same as in the Gaussian case.

Other methods to obtain multiscale representations of audio signals include wavelet transforms [75, 148]. Whereas the temporal resolution of

spectrograms is constant across the entire frequency range, wavelet representations have a higher temporal resolution for higher frequencies, so they inherently encompass multiple timescales. Anden et al. [3] have extended MFCC features using wavelet filter banks to allow for non-stationary spectral information to be captured. Other work on multiscale representations of music includes that of Foucard et al. [41], who used boosting to combine features expressed at different timescales. Finally, Morgan et al. [105] have investigated the use of multiple analysis windows with different lengths in the context of speech recognition, akin to the multiresolution spectrograms approach, to increase robustness against noise.

3.5 Proposed approach

Recent results in feature learning indicate that simple techniques can be quite effective, provided that the learnt feature representations are sufficiently high-dimensional. In particular, the K-means algorithm turns out to be very suitable for unsupervised feature learning, sometimes surpassing more complicated approaches based on restricted Boltzmann machines (RBMs), autoencoders and sparse coding in terms of discriminative performance [28].

Using the K-means algorithm for feature learning has several advantages: it is simple, well-known and easy to implement. It has only a single hyperparameter (the number of centroids, corresponding to the size of the resulting feature representation), and is typically several orders of magnitude faster to train than RBMs or autoencoders. This also eliminates the need to invest in special hardware, since it runs fast enough on commodity CPUs. It has previously been applied to spectrograms to learn features for genre classification [158] and for music similarity estimation [125].

Coates and Ng[27] describe in detail how K-means can be used for feature learning, and suggest to use a modified version where the centroids are constrained to have a unit L2 norm (they must lie on the unit sphere). This is achieved by adding a normalization step in every iteration after the means are updated. This algorithm is called *spherical K-means*.

I will extract multiscale time-frequency representations from the audio signals with up to 6 levels (numbered 0 to 5), as described in Section 3.4. I will then learn features on each level separately using the spherical K-means algorithm. To arrive at a multiscale feature representation, I will simply concatenate feature vectors obtained from each level into one large feature vector.

For each level, I will use the following feature extraction pipeline, which is visualised schematically in Figure 3.2. First, I divide the spectrograms into

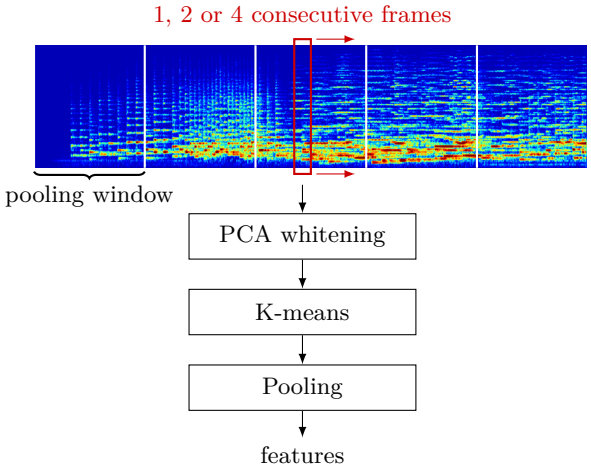


Figure 3.2: Schematic overview of the feature extraction process at a single timescale. The mel-spectrogram is divided into large pooling windows. Smaller windows consisting of 1, 2 or 4 consecutive frames are extracted convolutionally and PCA-whitened, and then K-means features are extracted. The features are pooled by taking the maximum across time over the pooling windows.

large pooling windows, several seconds in length. Smaller windows consisting of 1, 2 or 4 consecutive frames are then extracted convolutionally[2] and PCA-whitened, and spherical K-means features are extracted from the whitened windows. The features are pooled by taking the maximum across time over the pooling windows. The rest of this section describes each part of the pipeline in detail.

Frames taken from multiresolution spectrograms will automatically model longer-range temporal structure at higher levels, because the spectrogram window size is increased. For the Gaussian and Laplacian pyramids however, this is not the case: any temporal structure present at longer timescales is lost by the downsampling operation that is required to construct the higher levels of the pyramid. Although frames at higher levels of the pyramid are affected by a larger region of the input, they do not reflect temporal structure within this region as a result. This is why I use windows of a small number of consecutive frames as input instead: it allows for this structure to be taken into account by the feature learning algorithm and improves discriminative performance [111].

[2]This means that the extracted windows overlap by all but one frame.

3.5.1 Time-frequency representation

We will use log-scaled mel-spectrograms as a time-frequency representation. The mel frequency scale is used rather than a linear one, because it matches more closely how humans perceive pitch and it reduces the dimensionality of the input. Other possible representations include constant-Q spectrograms, where the frequency scale is logarithmic, and auditory spectrograms, which are based on the properties of the human auditory system [94]. Although I opted to use log-scaled mel-spectrograms for our experiments, the feature learning approach is generic and not dependent on the type of time-frequency representation that is used.

For all experiments described in this paper, I extracted spectrograms with a window size of 1024 audio samples and a hop size of 512 samples. These spectrograms have a linear frequency scale, so I converted them to mel-spectrograms with 128 components by pooling components together. I converted the mel-spectrograms to a logarithmic intensity scale by applying the elementwise function $f(x) = \log(1 + C \cdot x)$, where C is a constant controlling the amount of compression [106], which I set to $10,000$ for the best results.

3.5.2 PCA whitening

Before applying the feature learning algorithm, the input is first whitened using PCA. This significantly improves the features learnt by the K-means algorithm [28]. In practice, I randomly sample a set of $100,000$ windows from the data to compute the whitening transform. Enough components are kept to retain 99% of the variance. The whitened windows are then used to learn the dictionary.

3.5.3 Spherical K-means

K-means has often been used as a dictionary learning algorithm in the past, but it has only recently been shown to be competitive with more advanced techniques such as sparse coding. The one-of-K coding used in the algorithm (i.e., each example being assigned to a single mean) is beneficial during learning, but it turns out to be less suitable for the encoding phase [26]. By replacing the encoding procedure, the features become significantly more useful. For spherical K-means, a linear encoding scheme works well: the feature representation of a data point is obtained by multiplying it with the dictionary matrix: this is the concatenation of all the learned mean vectors in the original space. By performing this matrix multiplication, we compute the dot product between the data point and each of the mean vectors. The

resulting scalars measure the similarity between the mean vectors and the data point, and together they form its feature representation.

To extract features from mel-spectrograms with a sliding window, we have to whiten the windows and extract features, which can be implemented as a single convolution with the product of the whitening matrix and the dictionary matrix. This makes the feature extraction step efficient and simple to implement.

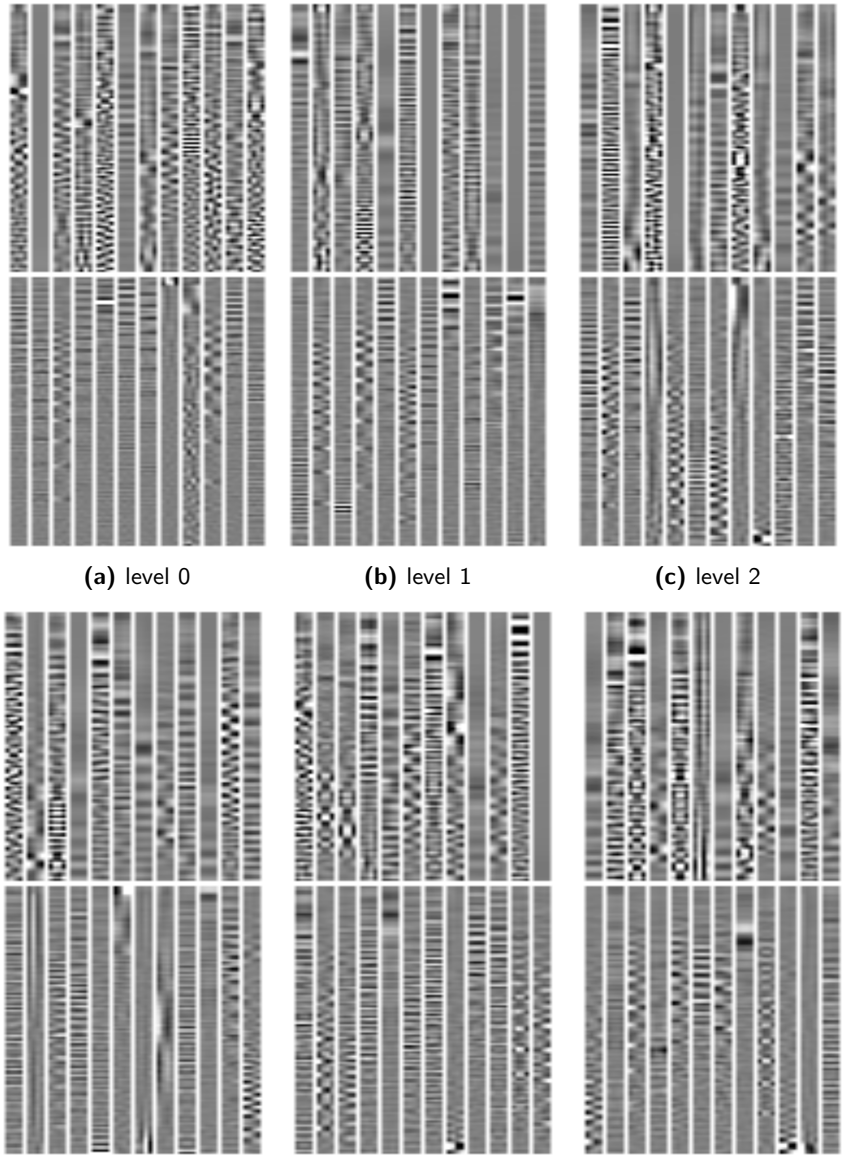
The K-means step can also be skipped altogether and the PCA components obtained after whitening can be used as features directly instead. These features were referred to as principal mel-spectrum components (PMSCs) by Hamel et al. [53]. They are in fact very similar to MFCCs: if the windows consist of single frames, replacing the PCA whitening step with a discrete cosine transform (DCT) results in MFCC vectors. Both transformations serve to decorrelate the input.

The feature learning algorithm is applied to windows sampled from the data, and as a result, the features it learns are expected to reconstruct entire windows. This is sometimes termed *patch-based training* [71]. Nevertheless, these features are then extracted from the data in a convolutional fashion, leading to a certain redundancy: groups of features may be learnt that differ only by a translation in the time direction. This can be prevented by using *convolutional training* instead, by allowing the features to be translated when reconstructing the windows during training as well. However, this is much more computationally demanding, and adapting the feature learning algorithm in this way is not straightforward. In the end, the simplicity of the patch-based approach trumps the effects of having multiple redundant features, and a simpler solution to this problem is to learn a larger number of features instead.

Figure 3.3 shows a random selection of features learnt from windows of 4 consecutive mel-spectrogram frames at different levels in a 6-level Gaussian pyramid, with PCA whitening (top) and with spherical K-means (bottom). Some features are stationary across the time dimension, others resemble percussive events and changes in pitch. Many of the K-means features seem to reflect harmonic structure. This is especially the case for the two lowest levels, where the features span between 160 and 320 ms. This type of structure is less pronounced in the PCA features.

3.5.4 Pooling

The representation we obtain by extracting features convolutionally from mel-spectrograms is very large, and for many tasks, it is beneficial to summarise the features across time first. The features are pooled across large time windows several seconds in length. Although a combination of the



(a) level 0 **(b)** level 1 **(c)** level 2

(d) level 3 **(e)** level 4 **(f)** level 5

Figure 3.3: A random selection of features learnt with PCA whitening (top) and spherical K-means (bottom) from windows of 4 consecutive mel-spectrogram frames sampled from the Magnatagatune dataset, at different levels in a Gaussian pyramid. The frequency increases from bottom to top.

mean, variance, minimum and maximum across time has been found to work well for this purpose [53], we use only the maximum, because it was found to be the best performing single pooling function. This reduces the size of the feature representation fourfold, which speeds up experiments significantly while having only a limited impact on performance.

I used non-overlapping pooling windows of 128 frames. Because the time resolution of the higher levels of the pyramid representations is coarser, the pooling window has to be shrunk accordingly so that the feature vectors from different levels are aligned and can be concatenated.

A result of the pooling operation is that accurate time information is lost: we only retain information about which features are active for a given input, but not when in the audio signal they occur. For the tasks investigated in this paper (see Section 3.6) this information is not necessary.

Note that without this pooling stage, there would be a linear relation between features extracted using Gaussian and Laplacian pyramids. The non-linear operation of taking the maximum across time ensures that we get different feature vectors for both representations. If we had used a linear pooling function such as the mean, the resulting feature vectors would be linear transformations of each other, so they would contain exactly the same information.

3.5.5 Frequency invariance

Salient patterns in music tend to occur at many different frequencies. For example, a song often features many pitch-shifted versions of the same motifs. For many tasks, the precise frequencies at which these patterns occur are not important, so it is useful to have a feature representation that is frequency invariant. We can modify the proposed feature extraction pipeline to incorporate frequency invariance, by introducing a pooling step in the frequency direction as well as in the time direction. The features will then span fewer than 128 mel-spectrogram components. I will also use the maximum function for frequency pooling.

3.5.6 Feature hierarchies

Another possible modification is to extract a hierarchy of features with multiple layers. I will simply apply the combination of PCA whitening and K-means feature learning a number of times in succession, to extract higher-level features. Note that due to the linearity of the convolutional feature extraction (see Section 3.5.3), higher-layer features will be linear combinations of lower-layer features. Because we are working with multiscale representations of audio signals, it is important to distinguish the layers of the feature

hierarchy from the levels of the multiple timescale representations. We will extract multiple layers of features separately for each timescale.

3.6 Experiments

Four different feature learning setups were evaluated: PCA whitening, and spherical K-means with 200, 500 and 1000 means. I also compared 7 different multiscale approaches: Gaussian and Laplacian pyramids with windows of 1, 2 and 4 consecutive frames, and multiresolution spectrograms. This yields 28 different architectures in total. I used 4 tasks on 3 datasets to evaluate these architectures: tag prediction and similarity metric learning on Magnatagatune, tag prediction on the Million Song Dataset and genre recognition on GTZAN. The extensions proposed in Sections 3.5.5 and 3.5.6 were evaluated on the Magnatagatune tag prediction task only.

3.6.1 Datasets

- **Magnatagatune**: the Magnatagatune dataset [81] contains 25 863 29-second audio clips taken from songs by 230 artists sampled at 16 000 Hz, along with metadata and tags. It comes in 16 parts, of which I used the first 12 for training, the 13th for validation and the remaining 3 for testing.

- **GTZAN**: the GTZAN music genre dataset [147] is one of the most frequently used datasets in MIR. It contains 1000 30-second excerpts, sampled at 22 050 Hz, categorised into 10 genres. Although some problems with this dataset have been pointed out recently [143], I still include it for comparison purposes. Due to its small size, I report results obtained with 5-fold cross-validation.

- **Million Song Dataset**: the Million Song Dataset [18] is a collection of metadata and precomputed audio features for one million contemporary songs. Although no audio excerpts are provided, the metadata includes unique identifiers for most of the songs, with which 29 second audio clips can be downloaded from 7digital.com. In this way, I obtained audio clips for 10 000 songs sampled at 22 050 Hz. I used 7000 for training, 1000 for validation and 2000 for testing.

3.6.2 Tag prediction

The tag prediction task allows us to evaluate the versatility of the representations, because tags describe a variety of different aspects of the music: genre, instrumentation, tempo and dynamics, among others. All clips in the Magnatagatune dataset are annotated with tags from a set of 188. I only used the 50 most frequently occurring tags for our experiments, to ensure that enough training data is available for each of them. For the Million Song Dataset, I used 50 of the most popular last.fm tags, excluding user-specific tags such as 'favorites', 'awesome' and 'seen live'.

I trained a multilayer perceptron (MLP) on the proposed multiscale feature representations to predict the presence of the tags, with a hidden layer consisting of 1000 rectified linear units [110]. I used minibatch gradient descent with weight decay to minimise the cross entropy between the predictions and the true tag assignments, and stopped training when the performance on the validation set was no longer increasing. The hyperparameters I used are listed in Table 3.1. I trained the MLP on feature vectors obtained from pooling windows. Tag predictions for an entire clip were computed by taking the average of the predictions across all pooling windows. I computed the area under the ROC-curve (AUC) for all tags individually and then took the average across all tags to get a measure of the predictive performance of the trained models.

I also used the tag prediction task on Magnatagatune to evaluate the model extensions proposed in Sections 3.5.5 and 3.5.6. For the frequency invariance experiment, I varied the amount of invariance (i.e. the size of the pooling window in the frequency dimension) from 1 to 32, to determine how much invariance is necessary. For the feature hierarchy experiment, I learnt three layers of features and compared the performance of the individual layers, as well as a combination of the first and second layer, and a combination of all three layers.

3.6.3 Similarity metric learning

The features were also used to learn a music similarity metric on the Magnatagatune dataset. Using Neighborhood Components Analysis (NCA) [47], a linear projection of the features into a 50-dimensional space is learnt, such that similar clips are close together in terms of Euclidean distance. Each clip is mapped into this space by projecting the feature vectors corresponding to each pooling window and then taking the mean across all pooling windows. The linear projection matrix is then optimised with minibatch gradient descent to project clips by a given artist into the same region. This approach was previously explored by Slaney et al. [136].

	Magnatagatune		MSD	GTZAN
	MLP	NCA	MLP	MLP
learning rate	0.1	0.01	0.1	0.003
weight decay	10^{-5}	10^{-3}	10^{-5}	0
minibatch size	100	100	100	100
min. number of epochs	100	100	100	-
max. number of epochs	500	500	500	-
epoch patience	10	20	10	-
number of epochs	-	-	-	100
number of hidden units	1000	-	1000	100
number of dimensions	-	50	-	-

Table 3.1: Hyperparameter values used to train the MLPs for tag prediction on Magnatagatune and the Million Song Dataset, and genre recognition on GTZAN, as well as NCA for similarity metric learning on Magnatagatune. The 'epoch patience' is the number of epochs to continue training when performance on the validation set has not increased.

NCA is based on a probabilistic version of K-nearest neighbor classification, where neighbors are selected probabilistically proportionally to their distance and each data point inherits the class of its selected neighbour. The objective is then to maximise the probability of correct classification. I report this probability on the test set. Both NCA and the MLP used for the tag prediction task were implemented using the Theano library [14].

3.6.4 Genre recognition

For genre recognition on GTZAN, I also trained an MLP for each of the architectures, this time with 100 hidden units and for 100 epochs (see Table 3.1). I trained on feature vectors obtained from pooling windows, and averaged predictions across pooling windows to obtain predictions for an entire clip. I measured the classification accuracy to assess the discriminative performance of the features.

3.7 Results

3.7.1 Architectures

The results for the tag prediction task on Magnatagatune and the Million Song Dataset obtained with each of the 28 different architectures are shown

in Figure 3.4 and Figure 3.5 respectively. All reported results are averaged across 10 MLP training runs with different initializations. Unfortunately I cannot directly compare the results with those of Hamel et al. [54], who evaluated a combination of multiresolution spectrograms with PCA whitening for feature learning, because they used a different version of the Magnatagatune dataset which is not publicly available.

Using features learnt with spherical K-means almost always yields increased performance, although the difference between using 500 or 1000 means is usually negligible. Interestingly, the best performing architecture for Magnatagatune uses a Laplacian pyramid, with features learnt from single frames. This is somewhat unexpected, because it implies that grouping consecutive frames into windows is not necessary for this type of multiscale representation. This does seem to help when using a Gaussian pyramid, however. A Gaussian pyramid with windows of 2 consecutive frames works well for both datasets.

For the similarity metric task on Magnatagatune, results are shown in Figure 3.6. A Gaussian pyramid with windows of 2 consecutive frames works best. Using 1000 means is noticeably worse than using 500 means. This can be attributed to the fact that the NCA objective is very prone to overfitting when using a large amount of input features (6000 in this case, for 6 levels), despite the use of weight decay and early stopping for regularization.

Results for the genre recognition task on GTZAN are shown in Figure 3.7. Here, the difference between using PCA and spherical K-means features is especially pronounced. A Gaussian pyramid with windows of 2 consecutive frames also works best for this task. We achieve an accuracy of 84.8%, which is comparable to the state of the art on this dataset.

3.7.2 Relevant timescales

To assess the merit of using a multiscale representation, and to determine which timescales are relevant for different tags in the tag prediction task, I took the best architecture from the tag prediction task on Magnatagatune and tried to predict tags from each level individually. Although a combination of all levels performs best for all tags, it is not always obvious precisely which timescales are the most relevant ones for a given tag. Figure 3.8 shows a selection of tags where some patterns can be identified.

Two tags describe the tempo of the music: *slow* and *fast*. Both tags benefit quite a lot from the multiscale representation: a combination of all levels performs much better than any level individually. Tags describing dynamics, such as *loud*, *quiet* and *soft*, seem to rely mostly on the top levels, corresponding to the coarsest timescales. This may also be because the top level is the only level in the Laplacian pyramid that is not a difference of

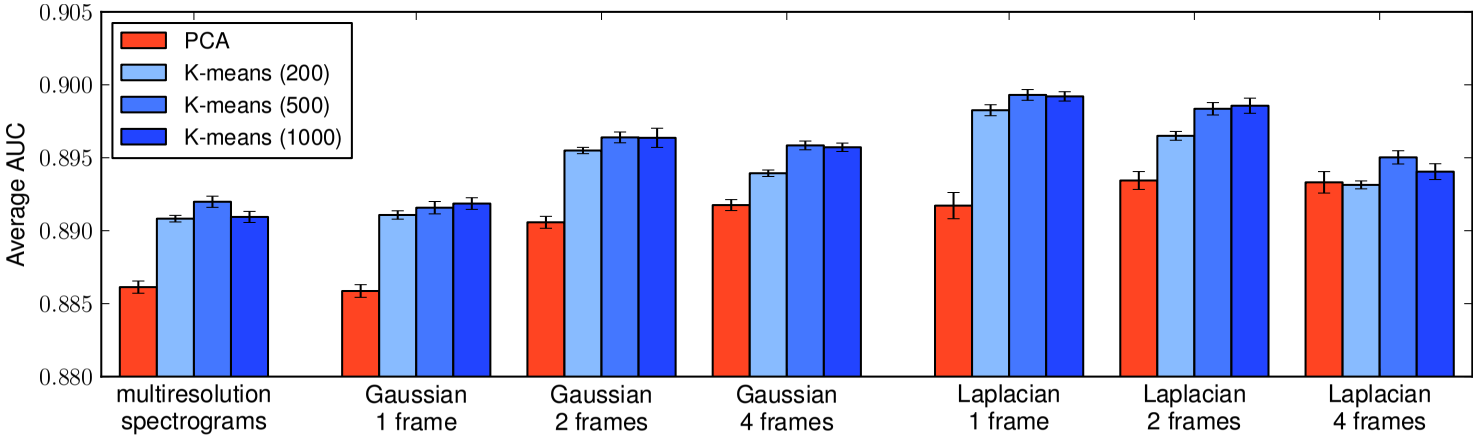


Figure 3.4: Results for the tag prediction task on Magnatagatune, for 28 different multiscale feature learning architectures. All reported results are averaged across 10 MLP training runs with different initializations. Error bars indicate the standard deviation across these 10 runs.

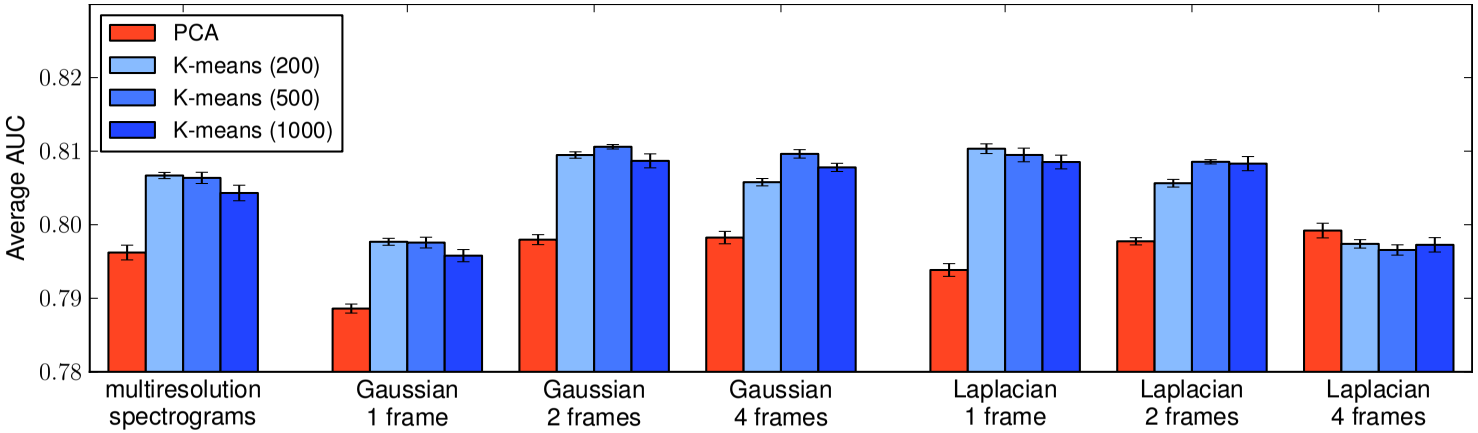


Figure 3.5: Results for the tag prediction task on the Million Song Dataset, for 28 different multiscale feature learning architectures. All reported results are averaged across 10 MLP training runs with different initializations. Error bars indicate the standard deviation across these 10 runs.

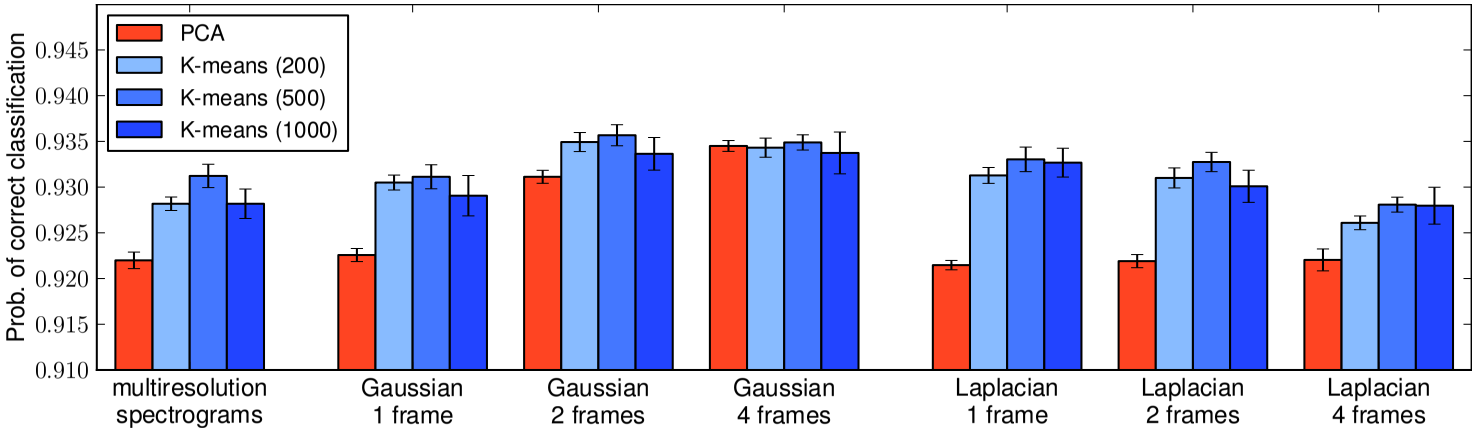


Figure 3.6: Results for the similarity metric learning task on Magnatagatune, for 28 different multiscale feature learning architectures. All reported results are averaged across 10 training runs with different initializations. Error bars indicate the standard deviation across these 10 runs.

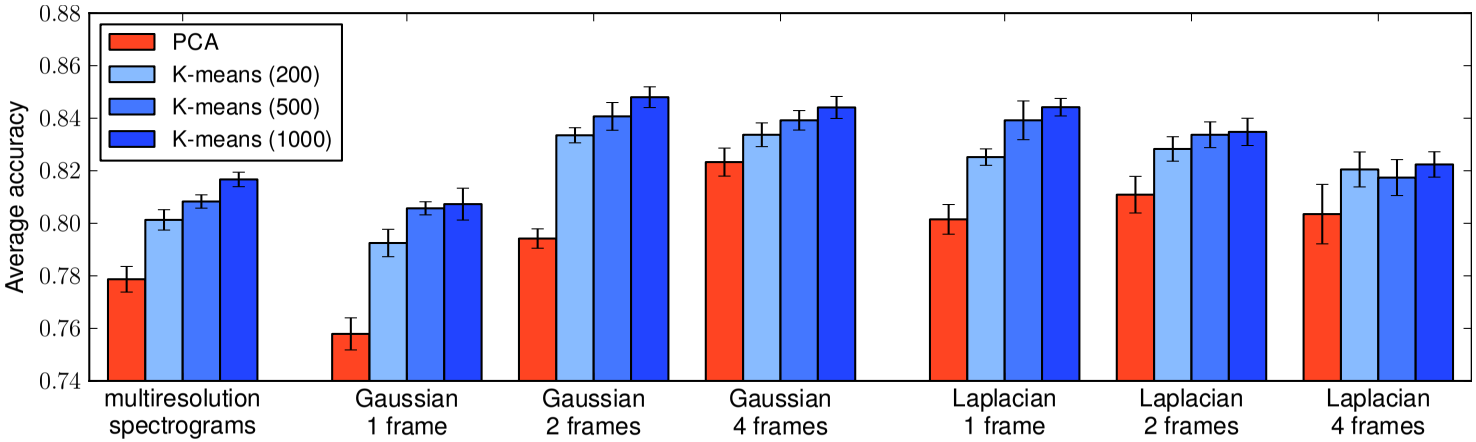


Figure 3.7: Results for the genre recognition task on GTZAN, for 28 different multiscale feature learning architectures. All reported results are obtained through 5-fold cross-validation and averaged across 10 training runs with different initializations. Error bars indicate the standard deviation across these 10 runs.

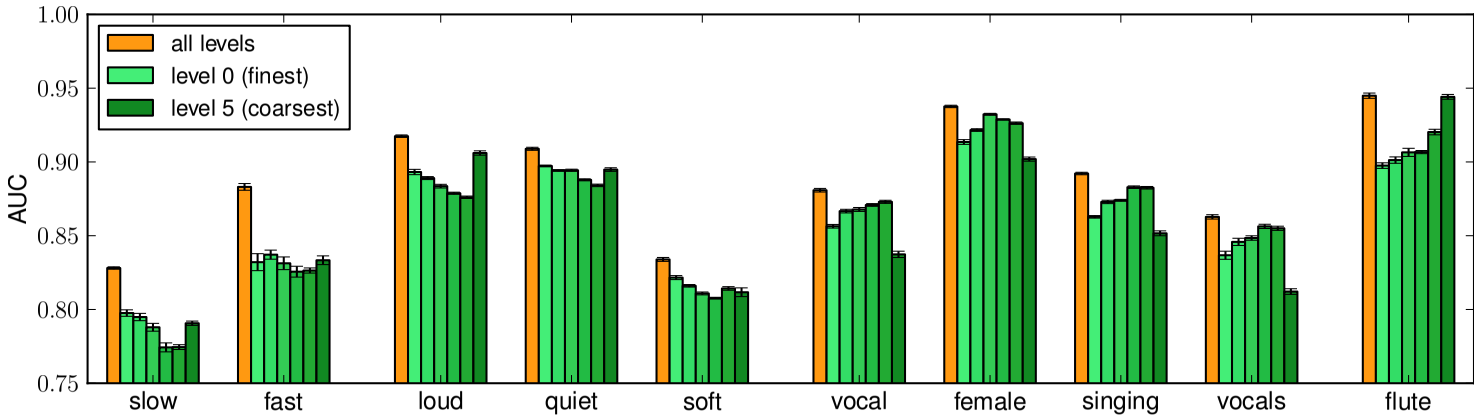


Figure 3.8: Results for a selection of individual tags, when using features from all levels combined and when using features from each level individually. The reported AUCs are for K-means with 500 means on the Laplacian pyramid with features learnt from single frames.

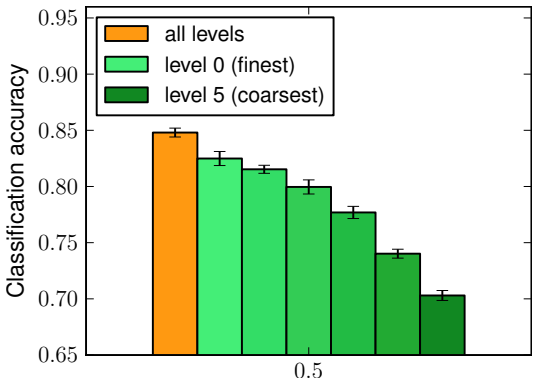


Figure 3.9: Results for the genre recognition task, when using features from all levels combined and when using features from each level individually. The reported accuracies are for K-means with 1000 means on the Gaussian pyramid with features learnt from pairs of consecutive frames.

two levels in the Gaussian pyramid.

Tags related to vocals can be predicted most accurately from intermediate levels, as evidenced by the results for *vocal*, *female*, *singing* and *vocals*. Of these, *female* is the easiest to predict, being the most specific tag. Finally, the *flute* tag is somewhat atypical among the tags describing instruments, in that it is the only one that relies mostly on the coarsest timescale (results for other instruments are not shown). A possible reason for this could be that the instrument lends itself well to playing longer, drawn out notes. A quick examination of the dataset reveals that many examples tagged *flute* feature such notes.

I conducted the same experiment for genre recognition (see Figure 3.9), and observed that the lower levels are clearly more important for this task. Nevertheless, using a combination of all levels still results in a noticeable increase in performance. Here, the reported accuracies are for K-means with 1000 means on the Gaussian pyramid with features learnt from pairs of consecutive frames.

3.7.3 Frequency invariance

Using the best performing architecture for the tag prediction task on Magnatagatune, I modified the pipeline to incorporate frequency invariance, as described in Section 3.5.5. The results are shown in Figure 3.10. It is clear that adding a limited amount of frequency invariance improves performance, from an average AUC of 0.8990 without frequency invariance to 0.9037 with a frequency invariance of 12. This increase is comparable to that obtained

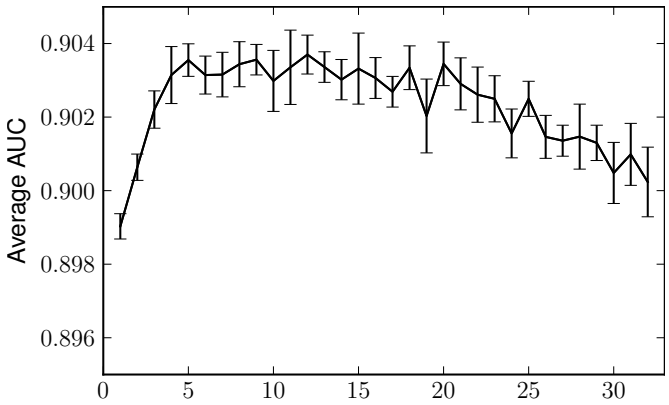


Figure 3.10: Results for the tag prediction task on Magnatagatune, for different levels of frequency invariance. The reported AUCs are for K-means with 500 means on the Laplacian pyramid with features learnt from single frames.

by using K-means for feature learning instead of using whitened PCA components as features directly. Performance increases rapidly initially as the amount of invariance is increased, after which it slowly decreases again.

3.7.4 Feature hierarchies

Using the same architecture, three layers of features were learnt, and I compared results using different combinations of layers, as shown in Figure 3.11 (see Section 3.5.6). All layers perform approximately the same individually. Combining layers results in a performance increase, albeit a very modest one. Presumably the linearity of the feature extraction limits the expressivity of the higher-layer features, so they do not provide much additional information.

3.8 Conclusion

I have proposed three approaches to building multiscale feature learning architectures for music audio, and I have evaluated them using several tasks designed to demonstrate the versatility of the learnt features. Learning features with the spherical K-means algorithm consistently improves results over just using PCA components, and a Gaussian pyramid representation with features learnt from pairs of consecutive frames seems to perform well for all tasks, although it is not always the best option. It is clear that learning features at multiple timescales improves performance over single-timescale

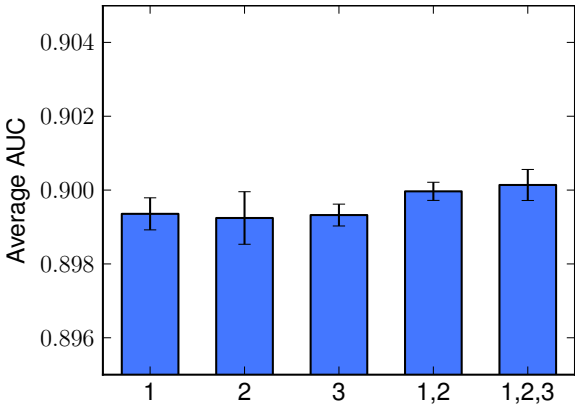


Figure 3.11: Results for the tag prediction task on Magnatagatune, for different combinations of feature layers. The reported AUCs are for K-means with 500 means on the Laplacian pyramid with features learnt from single frames.

approaches, and that the features perform very well in a discriminative setting, even though the learning algorithm is entirely unsupervised (only the classifier is learnt using label information). I also found that different kinds of tags tend to rely on different timescales.

The feature learning algorithm I used is very fast and easy to implement, making the proposed approach an accessible and efficient alternative to feature engineering. It is especially effective for tasks where large-scale temporal structure is of less importance. This is because features learnt at higher timescales are based on a much more coarse-grained representation of the audio signal. Although these features are able to capture more long-range structure, their resolution is much lower.

The feature learning method described in this chapter was originally introduced by Coates et al. [28], but it had only been applied to image data at that point. My success applying this method to time-frequency representations of audio signals inspired some researchers to do the same for different applications, including birdsong classification [142] and urban sound analysis [122].

I applied a single timescale version of the method with frequency pooling to the problem of detecting whale calls in underwater recordings in the context of a Kaggle competition: the 'Whale Detection Challenge'[3]. I spent only three days working on this competition. Nevertheless, using the proposed approach on spectrograms extracted from the recorded audio signals, I was able to finish 8th out of 245 participating teams. The fast training

[3]https://www.kaggle.com/c/whale-detection-challenge

procedure enabled me to try many different combinations of hyperparameters in a short timespan, which helped considerably to achieve a good rank. This is testament to the usefulness of the approach when the computational cost should be kept to a minimum.

The method makes some simplifying assumptions: the learnt features only capture a limited amount of long-range temporal structure, and what is captured is modeled at a coarse resolution. It also assumes that the tasks can be tackled without needing any exact temporal information about where certain patterns occur in the audio signal. As the MIR community moves on to more challenging tasks, these assumptions will become problematic and higher-capacity models that do not rely on them will be required. With today's hardware and the state of deep learning research, it is feasible to learn hierarchical representations on large collections of music audio using neural networks (both in a supervised and unsupervised fashion). In the long run I believe these models, which have fewer limitations and are better able to capture long-range structure and invariances, will be a more effective general purpose solution for a variety of MIR tasks.

4 Deep content-based music recommendation

4.1 Introduction

In previous chapters, I have primarily considered classification and tagging tasks. In practical applications of MIR, obtaining classifications or tag associations is not typically a goal in itself - this information is usually intended to serve some other purpose, such as facilitating music search or enabling content-based music recommendation. Tackling the intermediary task of assigning musical audio signals to one or more categories allows us to use many existing tools for this purpose: classification and tagging are paradigms that have been studied extensively, both outside and within MIR.

In this chapter, I will describe a different approach to content-based music recommendation, casting it as a regression problem instead. From data about the listening habits of a large number of users, *latent factor vectors* for songs are obtained using *collaborative filtering* techniques. These vectors of real numbers are compact but opaque representations of the various aspects of the music that affect listening preference. They obviate the need for a manually designed ontology of classes or tags that could be used as an alternative intermediate representation. A deep neural network model is then trained to learn to predict these vectors from audio signals. This allows us to incorporate content information directly into a more traditional recommendation approach based on historical listening data. This is especially useful for less popular music, where listening data is scarce. This biases the recommendations towards popular songs, which is undesirable. Content-based approaches have no such bias.

The work described in this chapter was done in close collaboration with my colleague Aäron van den Oord. I will use plural first person pronouns in the text to reflect this. My contribution consists primarily of the im-

plementation of the collaborative filtering algorithm we used to obtain the latent factor vectors (weighted matrix factorization [64]) and the extraction of spectrogram representations from the audio signals. Aäron designed and trained the convolutional neural networks and conducted the baseline experiments. I also wrote the majority of the paper resulting from this research, which was presented at NIPS 2013 [149].

The rest of this chapter is structured as follows: in Section 4.2, I describe different approaches to automatic music recommendation. Section 4.3 covers the dataset we used for this work, which is based chiefly on the Million Song Dataset. Section 4.4 describes the weighted matrix factorization (WMF) algorithm for collaborative filtering. Section 4.5 describes our approach and experimental setup. Experiments and results are given in Section 4.6. Section 4.7 lists some related work and we conclude in Section 4.8.

4.2 Music recommendation

In recent years, the music industry has shifted more and more towards digital distribution through online music stores and streaming services such as iTunes, Spotify, Grooveshark and Google Play. As a result, automatic music recommendation has become an increasingly relevant problem: it allows listeners to discover new music that matches their tastes, and enables online music stores to target their wares to the right audience.

Although recommender systems have been studied extensively, the problem of music recommendation in particular is complicated by the sheer variety of different styles and genres, as well as social and geographic factors that influence listener preferences. The number of different items that can be recommended is very large, especially when recommending individual songs. This number can be reduced by recommending albums or artists instead, but this is not always compatible with the intended use of the system (e.g. automatic playlist generation), and it disregards the fact that the repertoire of an artist is rarely homogenous: listeners may enjoy particular songs more than others.

Many recommender systems rely on usage patterns: the combinations of items that users have consumed or rated provide information about the users' preferences, and how the items relate to each other. This is the *collaborative filtering* approach. Another approach is to predict user preferences from item content and metadata.

The consensus is that collaborative filtering will generally outperform content-based recommendation [135]. However, it is only applicable when usage data is available. Collaborative filtering suffers from the *cold start*

problem: new items that have not been consumed before cannot be recommended. Additionally, items that are only of interest to a niche audience are more difficult to recommend because usage data is scarce. In many domains, and especially in music, they comprise the majority of the available items, because the users' consumption patterns follow a power law [24]. Content-based recommendation is not affected by these issues.

4.2.1 Content-based music recommendation

Music can be recommended based on available metadata: information such as the artist, album and year of release is usually known. Unfortunately this will lead to predictable recommendations. For example, recommending songs by artists that the user is known to enjoy is not particularly useful.

One can also attempt to recommend music that is perceptually similar to what the user has previously listened to, by measuring the similarity between audio signals [136, 125]. This approach is almost always applicable, but it also very computationally demanding. Furthermore, it requires the definition of a suitable similarity metric. Such metrics are often defined ad hoc, based on prior knowledge about music audio, and as a result they are not necessarily optimal for the task of music recommendation. Because of this, some researchers have used user preference data to tune similarity metrics [101, 139].

4.2.2 Collaborative filtering

Collaborative filtering methods can be neighborhood-based or model-based [119]. The former methods rely on a similarity measure between users or items: they recommend items consumed by other users with similar preferences, or similar items to the ones that the user has already consumed. Model-based methods on the other hand attempt to model latent characteristics of the users and items, which are usually represented as vectors of latent factors. Latent factor models have been very popular ever since their effectiveness was demonstrated for movie recommendation in the Netflix Prize [12].

4.2.3 The semantic gap

Latent factor vectors form a compact description of the different facets of users' tastes, and the corresponding characteristics of the items. To demonstrate this, we computed latent factors for a small set of usage data, and listed some artists whose songs have very positive and very negative values for each factor in Table 4.1. This representation is quite versatile and can be

	Artists with positive values	Artists with negative values
1	Justin Bieber, Alicia Keys, Maroon 5, John Mayer, Michael Bublé	The Kills, Interpol, Man Man, Beirut, the bird and the bee
2	Bonobo, Flying Lotus, Cut Copy, Chromeo, Boys Noize	Shinedown, Rise Against, Avenged Sevenfold, Nickelback, Flyleaf
3	Phoenix, Crystal Castles, Muse, Röyksopp, Paramore	Traveling Wilburys, Cat Stevens, Creedence Clearwater Revival, Van Halen, The Police

Table 4.1: Artists whose tracks have very positive and very negative values for three latent factors. The factors seem to discriminate between different styles, such as indie rock, electronic music and classic rock.

used for other applications besides recommendation, as we will show later (see Section 4.6.1). Since usage data is scarce for many songs, it is often impossible to reliably estimate these factor vectors. Therefore it would be useful to be able to predict them from music audio content.

As previously discussed in Section 1.1.1, there is a is a large *semantic gap* between the characteristics of a song that affect user preference, and the corresponding audio signal. Extracting high-level properties such as genre, mood, instrumentation and lyrical themes from audio signals requires powerful models that are capable of capturing the complex hierarchical structure of music. Additionally, some properties are impossible to obtain from audio signals alone, such as the popularity of the artist, their reputation and and their location. This semantic gap is precisely why collaborative filtering tends to yield better results than content-based music recommendation when historical listening data is available.

In this chapter, we strive to bridge the semantic gap in music by training deep convolutional neural networks to predict latent factors from music audio. We evaluate our approach on the Million Song Dataset, an industrial-scale dataset with audio excerpts of over 380,000 songs, and compare it with a more conventional approach using a bag-of-words feature representation for each song. We assess to what extent it is possible to extract characteristics that affect user preference directly from audio signals, and evaluate the predictions from our models in a music recommendation setting.

4.3 The dataset

The Million Song Dataset (MSD) [18], which was also used for the work in chapters 2 and 3, is a collection of metadata and precomputed audio

features for one million contemporary songs. Several other datasets linked to the MSD are also available, featuring lyrics, cover songs, tags and user listening data. This makes the dataset suitable for a wide range of different music information retrieval tasks. Two linked datasets are of interest for our experiments:

- The Echo Nest Taste Profile Subset provides play counts for over 380,000 songs in the MSD, gathered from 1 million users. The dataset was used in the Million Song Dataset challenge [102] in 2012.
- The Last.fm dataset provides a multitude of tags for over 500,000 songs.

Traditionally, research in music information retrieval (MIR) on large-scale datasets was limited to industry, because large collections of music audio cannot be published easily due to licensing issues. As discussed before in the previous two chapters, the authors of the MSD circumvented these issues by providing precomputed features instead of raw audio. Unfortunately, the audio features provided with the MSD are of limited use, and the process by which they were obtained is not very well documented. The feature set was extended by Rauber et al. [117], but the absence of raw audio data, or at least a mid-level representation, is still an issue. However, we were able to attain 29 second audio clips for over 99% of the dataset from 7digital.com.

Due to its size, the MSD allows for the music recommendation problem to be studied in a more realistic setting than was previously possible. It is also worth noting that the Taste Profile Subset is one of the largest collaborative filtering datasets that are publicly available today.

4.4 Weighted matrix factorization

The Taste Profile Subset contains play counts per song and per user, which is a form of *implicit feedback*. We know how many times the users have listened to each of the songs in the dataset, but they have not explicitly rated them. However, we can assume that users will probably listen to songs more often if they enjoy them. If a user has never listened to a song, this can have many causes: for example, they might not be aware of it, or they might expect not to enjoy it. This setting is not compatible with traditional matrix factorization algorithms, which are aimed at predicting ratings.

We used the *weighted matrix factorization* (WMF) algorithm, proposed by Hu et al. [64], to learn latent factor representations of all users and items in the Taste Profile Subset. This is a modified matrix factorization algorithm aimed at implicit feedback datasets. Let r_{ui} be the play count for

user u and song i. For each user-item pair, we define a preference variable p_{ui} and a confidence variable c_{ui} ($I(x)$ is the indicator function, α and ϵ are hyperparameters):

$$p_{ui} = I(r_{ui} > 0), \tag{4.1}$$
$$c_{ui} = 1 + \alpha \log(1 + \epsilon^{-1} r_{ui}). \tag{4.2}$$

The preference variable indicates whether user u has ever listened to song i. If it is 1, we will assume the user enjoys the song. The confidence variable measures how certain we are about this particular preference. It is a function of the play count, because songs with higher play counts are more likely to be preferred. If the song has never been played, the confidence variable will have a low value, because this is the least informative case.

The WMF objective function is given by:

$$\min_{x_\star, y_\star} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u ||x_u||^2 + \sum_i ||y_i||^2 \right), \tag{4.3}$$

where λ is a regularization parameter, x_u is the latent factor vector for user u, and y_i is the latent factor vector for song i. It consists of a confidence-weighted mean squared error term and an L2 regularization term. Note that the first sum ranges over all users and all songs: contrary to matrix factorization for rating prediction, where terms corresponding to user-item combinations for which no rating is available can be discarded, we have to take all possible combinations into account. As a result, using stochastic gradient descent for optimization is not practical for a dataset of this size. Hu et al. propose an efficient alternating least squares (ALS) optimization method, which we used instead.

4.5 Predicting latent factors from music audio

Predicting latent factors for a given song from the corresponding audio signal is a regression problem. It requires learning a function that maps a time series to a vector of real numbers. We evaluate two methods to achieve this: one follows the conventional approach in MIR by extracting local features from audio signals and aggregating them into a bag-of-words (BoW) representation, discarding any large-scale temporal structure in the process. Any

traditional regression technique can then be used to map this feature representation to the factors. The other method is to use a deep convolutional network, which is able to take into account large-scale temporal structure in its higher layers.

Latent factor vectors obtained by applying WMF to the available usage data are used as ground truth to train the prediction models. It should be noted that this approach is compatible with any type of latent factor model that is suitable for large implicit feedback datasets. We chose to use WMF because an efficient optimization procedure exists for it.

4.5.1 Bag-of-words representation

Many MIR systems rely on the following feature extraction pipeline to convert music audio signals into a fixed-size representation that can be used as input to a classifier or regressor [101, 155, 154, 40, 63]:

- **Extract MFCCs from the audio signals.** We computed 13 MFCCs from windows of 1024 audio frames, corresponding to 23 ms at a sampling rate of 22050 Hz, and a hop size of 512 samples. We also computed first and second order differences, yielding 39 coefficients in total.
- **Vector quantize the MFCCs.** We learned a dictionary of 4000 elements with the K-means algorithm and assigned all MFCC vectors to the closest mean.
- **Aggregate them into a bag-of-words representation.** For every song, we counted how many times each mean was selected. The resulting vector of counts is a bag-of-words feature representation of the song.

We then reduced the size of this representation using PCA (we kept enough components to retain 95% of the variance) and used two different models on top of this to predict latent factors: linear regression and a multilayer perceptron with 1000 hidden units. We also used it as input for the metric learning to rank (MLR) algorithm [100], to learn a similarity metric for content-based recommendation. This was used as a baseline for our music recommendation experiments, which are described in Section 4.6.2.

4.5.2 Convolutional neural networks

Convolutional neural networks (CNNs) have recently been used to improve on the state of the art in speech recognition and large-scale image classification by a large margin [58, 73]. Three ingredients seem to be central to the success of this approach:

- Using rectified linear units (ReLUs) [110] instead of sigmoid nonlinearities leads to faster convergence and reduces the vanishing gradient problem that plagues traditional neural networks with many layers.

- Parallellization is used to speed up training, so that larger models can be trained in a reasonable amount of time. We used the Theano library [14] to take advantage of GPU acceleration.

- A large amount of training data is required to be able to fit large models with many parameters. The MSD contains enough training data to be able to train large models effectively.

We have also evaluated the use of dropout regularization [57], but this did not yield any significant improvements.

We first extracted an intermediate time-frequency representation from the audio signals to use as input to the network. We used log-compressed mel-spectrograms with 128 components and the same window size and hop size that we used for the MFCCs (1024 and 512 audio frames respectively). The networks were trained on windows of 3 seconds sampled randomly from the audio clips. This was done primarily to speed up training. To predict the latent factors for an entire clip, we averaged over the predictions for consecutive windows.

The network consisted of two convolutional layers with 128 filters spanning 6 timesteps and the entire frequency range, yielding one-dimensional convolutions. A max-pooling layer with a pooling window of 4 timesteps followed the first convolutional layer, and one with a pooling window of 5 timesteps followed the second convolutional layer. A fully connected hidden layer with 400 units was stacked on top of this, and finally the output layer had a number of units equaling the number of latent factors to predict (50 or 400, see Section 4.6.2).

Convolutional neural networks are especially suited for predicting latent factors from music audio, because they allow for intermediate features to be shared between different factors, and because their hierarchical structure consisting of alternating feature extraction layers and pooling layers allows them to operate on multiple timescales.

4.5.3 Objective functions

Latent factor vectors are real-valued, so the most straightforward objective is to minimize the mean squared error (MSE) of the predictions. Alternatively, we can also continue to minimize the weighted prediction error (WPE) from the WMF objective function. Let y_i be the latent factor vector for song i,

obtained with WMF, and y'_i the corresponding prediction by the model. The objective functions are then (θ represents the model parameters):

$$\min_{\theta} \sum_{i} ||y_i - y'_i||^2, \tag{4.4}$$

$$\min_{\theta} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y'_i)^2. \tag{4.5}$$

4.6 Experiments

4.6.1 Versatility of the latent factor representation

To demonstrate the versatility of the latent factor vectors, we compared them with audio features in a tag prediction task. Tags can describe a wide range of different aspects of the songs, such as genre, instrumentation, tempo, mood and year of release.

We ran WMF to obtain 50-dimensional latent factor vectors on a subset of 9,330 of the most popular songs in our dataset, and trained a logistic regression model to predict the 50 most popular tags from the Last.fm dataset for each song. We also trained a logistic regression model on a bag-of-words representation of the audio signals, which was first reduced in size using PCA (see Section 4.5.1). We used 10-fold cross-validation and computed the average area under the ROC curve (AUC) across all tags. This resulted in an average AUC of **0.69365** for audio-based prediction, and **0.86703** for prediction based on the latent factor vectors.

4.6.2 Latent factor prediction: quantitative evaluation

To assess quantitatively how well we can predict latent factors from music audio, we used the predictions from our models for music recommendation. For every user u and for every song i in the test set, we computed the score $x_u^T y_i$, and recommended the songs with the highest scores first. As mentioned before, we also learned a song similarity metric on the bag-of-words representation using metric learning to rank. In this case, scores for a given user are computed by averaging similarity scores across all the songs that the user has listened to.

The following four models were used to predict latent factor vectors:

Model	mAP	AUC
MLR	0.01801	0.60608
linear regression	0.02389	0.63518
MLP	0.025364	0.646112
CNN with MSE	0.05016	0.70987
CNN with WPE	0.04323	0.70101

Table 4.2: Results for all considered models on a subset of the dataset containing only the 9,330 most popular songs, and listening data for 20,000 users.

- Linear regression trained on the bag-of-words representation described in Section 4.5.1.
- A multi-layer perceptron (MLP) trained on the same bag-of-words representation.
- A convolutional neural network trained on log-scaled mel-spectrograms to minimize the mean squared error (MSE) of the predictions.
- The same convolutional neural network, trained to minimize the weighted prediction error (WPE) from the WMF objective instead.

For our initial experiments, we used a subset of the dataset containing only the 9,330 most popular songs, and listening data for only 20,000 users. We used 1,881 songs for testing, leaving 7,449 songs for training. For the other experiments, we used all available data: we used all songs that we have usage data for and that we were able to download an audio clip for (382,410 songs and 1 million users in total, 46,728 songs were used for testing, leaving 335,682 songs for training).

Model	mAP	AUC
random	0.00015	0.49935
linear regression	0.00101	0.64522
CNN with MSE	0.00672	0.77192
upper bound	0.23278	0.96070

Table 4.3: Results for linear regression on a bag-of-words representation of the audio signals, and a convolutional neural network trained with the MSE objective, on the full dataset (382,410 songs and 1 million users). Also shown are the scores achieved when the latent factor vectors are randomized, and when they are learned from usage data using WMF (upper bound).

We report the mean average precision (mAP, cut off at 500 recommendations per user) and the area under the ROC curve (AUC) of the predictions. We evaluated all models on the subset, using latent factor vectors with 50 dimensions. We compared the convolutional neural network with linear regression on the bag-of-words representation on the full dataset as well, using latent factor vectors with 400 dimensions. Results are shown in Tables 4.2 and 4.3 respectively.

On the subset, predicting the latent factors seems to outperform the metric learning approach. Using an MLP instead of linear regression results in a slight improvement, but the limitation here is clearly the bag-of-words feature representation. Using a convolutional neural network results in another large increase in performance. Most likely this is because the bag-of-words representation does not reflect any kind of temporal structure.

Interestingly, the WPE objective does not result in improved performance. Presumably this is because the weighting causes the importance of the songs to be proportional to their popularity. In other words, the model will be encouraged to predict latent factor vectors for popular songs from the training set very well, at the expense of all other songs.

On the full dataset, the difference between the bag-of-words approach and the convolutional neural network is much more pronounced. Note that we decided not to train an MLP on this dataset due to the small difference in performance with linear regression on the subset. We also included results for when the latent factor vectors are obtained from usage data. This is an upper bound to what is achievable when predicting them from content. There is a large gap between our best result and this theoretical maximum, but this is to be expected: as we mentioned before, many aspects of the songs that influence user preference cannot possibly be extracted from audio signals only. In particular, we are unable to predict the popularity of the songs, which considerably affects the AUC and mAP scores.

4.6.3 Latent factor prediction: qualitative evaluation

Evaluating recommender systems is a complex matter, and accuracy metrics by themselves do not provide enough insight into whether the recommendations are sound. To establish this, we also performed some qualitative experiments on the subset. For each song, we searched for similar songs by measuring the cosine similarity between the predicted usage patterns. We compared the usage patterns predicted using the latent factors obtained with WMF (50 dimensions), with those using latent factors predicted with a convolutional neural network. A few songs and their closest matches ac-

cording to both models are shown in Table 4.4. When the predicted latent factors are used, the matches are mostly different, but the results are quite reasonable in the sense that the matched songs are likely to appeal to the same audience. Furthermore, they seem to be a bit more varied, which is a useful property for recommender systems.

Following McFee et al. [101], we also visualized the distribution of predicted usage patterns in two dimensions using t-SNE [150]. A few close-ups are shown in Figure 4.1. Clusters of songs that appeal to the same audience seem to be preserved quite well, even though the latent factor vectors for all songs were predicted from audio.

4.7 Related work

Many researchers have attempted to mitigate the cold start problem in collaborative filtering by incorporating content-based features. We review some recent work in this area of research.

Wang et al. [152] extend probabilistic matrix factorization (PMF) [121] with a topic model prior on the latent factor vectors of the items, and apply this model to scientific article recommendation. Topic proportions obtained from the content of the articles are used instead of latent factors when no usage data is available. The entire system is trained jointly, allowing the topic model and the latent space learned by matrix factorization to adapt to each other. Our approach is sequential instead: we first obtain latent factor vectors for songs for which usage data is available, and use these to train a regression model. Because we reduce the incorporation of content information to a regression problem, we are able to use a deep convolutional network.

Stenzel et al. [139] cluster playlists of songs in an unsupervised fashion, and use the affinities of songs with these clusters as a form of latent representation. They then compute cosine similarities between these representations to find similar songs. To incorporate content, they train an SVM to predict these affinities from 30 audio features extracted using the Marsyas framework [147]. They observe that recommendations improve significantly over using a purely content-based approach (i.e. defining a similarity measure directly on the audio features).

McFee et al. [101] define an artist-level content-based similarity measure for music learned from a sample of collaborative filter data using metric learning to rank [100]. They use a variation on the typical bag-of-words approach for audio feature extraction (see Section 4.5.1). Their results corroborate that relying on usage data to train the model improves content-based

Query	Most similar tracks (WMF)	Most similar tracks (predicted)
Jonas Brothers - Hold On	Jonas Brothers - Games Miley Cyrus - G.N.O. (Girl's Night Out) Miley Cyrus - Girls Just Wanna Have Fun Jonas Brothers - Year 3000 Jonas Brothers - BB Good	Jonas Brothers - Video Girl Jonas Brothers - Games New Found Glory - My Friends Over You My Chemical Romance - Thank You For The Venom My Chemical Romance - Teenagers
Beyoncé - Speechless	Beyoncé - Gift From Virgo Beyonce - Daddy Rihanna / J-Status - Crazy Little Thing Called Love Beyoncé - Dangerously In Love Rihanna - Haunted	Daniel Bedingfield - If You're Not The One Rihanna - Haunted Alejandro Sanz - Siempre Es De Noche Madonna - Miles Away Lil Wayne / Shanell - American Star
Coldplay - I Ran Away	Coldplay - Careful Where You Stand Coldplay - The Goldrush Coldplay - X & Y Coldplay - Square One Jonas Brothers - BB Good	Arcade Fire - Keep The Car Running M83 - You Appearing Angus & Julia Stone - Hollywood Bon Iver - Creature Fear Coldplay - The Goldrush
Daft Punk - Rock'n Roll	Daft Punk - Short Circuit Daft Punk - Nightvision Daft Punk - Too Long (Gonzales Version) Daft Punk - Aerodynamite Daft Punk - One More Time / Aerodynamic	Boys Noize - Shine Shine Boys Noize - Lava Lava Flying Lotus - Pet Monster Shotglass LCD Soundsystem - One Touch Justice - One Minute To Midnight

Table 4.4: A few songs and their closest matches in terms of usage patterns, using latent factors obtained with WMF and using latent factors predicted by a convolutional neural network.

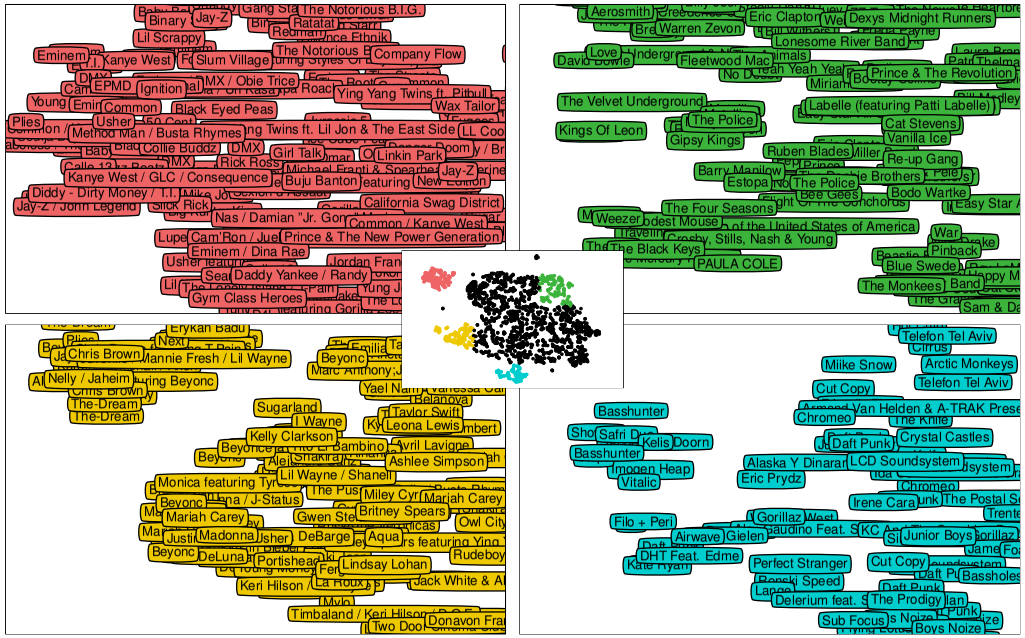


Figure 4.1: t-SNE visualization of the distribution of predicted usage patterns, using latent factors predicted from audio. A few close-ups show artists whose songs are projected in specific areas. We can discern hip-hop (red), rock (green), pop (yellow) and electronic music (blue). This figure is best viewed in color.

recommendations. For audio data they used the CAL10K dataset, which consists of 10,832 songs, so it is comparable in size to the subset of the MSD that we used for our initial experiments.

Weston et al. [155] investigate the problem of recommending items to a user given another item as a query, which they call 'collaborative retrieval'. They optimize an item scoring function using a ranking loss and describe a variant of their method that allows for content features to be incorporated. They also use the bag-of-words approach to extract audio features and evaluate this method on a large proprietary dataset. They find that combining collaborative filtering and content-based information does not improve the accuracy of the recommendations over collaborative filtering alone.

Both McFee et al. and Weston et al. optimized their models using a ranking loss. We have opted to use quadratic loss functions instead, because we found their optimization to be more easily scalable. Using a ranking loss instead is an interesting direction of future research, although we suspect that this approach may suffer from the same problems as the WPE objective (i.e. popular songs will have an unfair advantage).

4.8 Conclusion

We have investigated the use of deep convolutional neural networks to predict latent factors from music audio when they cannot be obtained from usage data. We evaluated the predictions by using them for music recommendation on an industrial-scale dataset. Even though a lot of characteristics of songs that affect user preference cannot be predicted from audio signals, the resulting recommendations seem to be sensible. We can conclude that predicting latent factors from music audio is a viable method for recommending new and less well-known music.

We also showed that recent advances in deep learning translate very well to the music recommendation setting in combination with this approach, with deep convolutional neural networks significantly outperforming a more traditional approach using bag-of-words representations of audio signals. This bag-of-words representation is used very often in MIR, and our results indicate that a lot of research in this domain could benefit significantly from using deep neural networks. This is not surprising, given the impressive improvements that have been achieved with these methods in other domains such as computer vision and speech recognition.

As mentioned in the introduction of this chapter, this work was presented at NIPS 2013. It was accompanied by an interactive demo: people could suggest any YouTube video of their choosing, an excerpt of which was then

analysed by our convolutional neural network. Ten samples from the Million Song Dataset whose predicted latent factors were closest to the factors of the analyzed excerpt where then presented as recommendations[1]. This allowed conference attendees to evaluate our approach qualitatively for themselves.

We made a conscious decision to present this work at NIPS 2013, a general machine learning conference, instead of a venue targeted at MIR researchers. Our intention was to showcase MIR as an interesting application among the broader deep learning community. Until now, most research into the application of deep learning for MIR problems has been internal to the MIR community.

A few authors have expanded on our work. Liang et al. [89] explored using a neural network trained on audio content as a prior in a latent factor model, integrating information from content and collaborative filtering. Wang and Wang [153] similarly attempted to integrate both content and collaborative filtering information into a single probabilistic model, although they did not use a convolutional architecture to model the audio content.

There has been considerable interest in our work from industry as well. Following the publication of our paper, both Aäron and I were offered internships to work on music recommendation, at Google and Spotify respectively. At Spotify, I expanded upon the work described in this chapter. I was able to apply it on a larger dataset, consisting of the 1 million most popular tracks on Spotify at the time. Multiple collaborative filtering models are used in their recommendation pipeline, so I used the latent factors computed by these algorithms as targets. I also scaled up the convolutional neural network to have more layers and more units per layer, as the model used in this work is relatively small by modern standards. I employed data augmentation and dropout regularization to reduce overfitting due to the larger number of trainable parameters. I have described my work at Spotify in some detail in a blog post[2].

[1] Note that we excluded all training data to avoid issues resulting from overfitting.
[2] http://benanne.github.io/2014/08/05/spotify-cnns.html

5 End-to-end learning

5.1 Introduction

Feature learning and deep learning have become the gold standard in several research domains that were previously dominated by manual feature design. Most notably, these domains include computer vision and speech recognition, where feature learning techniques have advanced the state of the art by a significant margin [58, 73]. This has allowed researchers to build models of data requiring only a minimum of prior knowledge.

As indicated in previous chapters, feature learning has also been receiving more attention from the MIR community lately [67]. However, most research on the application of feature learning to MIR problems relies on using mid-level representations of audio, such as spectrograms. This is also a form of prior knowledge. Some researchers also use handcrafted feature representations as input to feature learning algorithms (I did the same in chapter 2). In computer vision, on the other hand, modern feature learning techniques are able to operate directly on raw pixel representations of images, without requiring any form of preprocessing or feature extraction on beforehand [82].

In this chapter, I investigate whether it is possible to apply feature learning directly to raw audio signals, thus further reducing the amount of prior knowledge required when using feature learning for MIR tasks. I train convolutional neural networks to perform an automatic tagging task and compare different input representations and network architectures.

This chapter is structured as follows: in Section 5.2, I describe the mid-level representations of audio that are frequently used in MIR. In Section 5.3, I discuss the end-to-end learning paradigm. Experiments and results are described in Section 5.4, and I draw conclusions in Section 5.5.

5.2 Mid-level representations

Many higher-level characteristics of sound relate to the energies in different frequency bands. This explains the utility of time-frequency representations of audio such as spectrograms, which are frequently used in literature [85, 56, 158, 32] (see Section 1.1.2). Another advantage in the context of feature learning is that they convert the audio data into a representation that is very similar to an image. This makes it easier to translate feature learning approaches, which are often designed with image data in mind, to an audio context.

Most researchers do not use raw spectrograms, which have a linear frequency scale. The mel scale or a logarithmic frequency scale are preferred instead: they reduce the resolution for higher frequencies, which matches the human perception of frequency, and reduces the size of the representation (and hence computational requirements). Another common preprocessing step is to apply some form of dynamic range compression, for example by taking the logarithm of the spectrograms.

Although there has been some work about learning features directly from speech signal fragments [86, 87, 69], to our knowledge feature learning has not been applied directly to raw music audio signals in literature.

5.3 End-to-end learning

The term *end-to-end learning* is used to refer to processing architectures where the entire stack, connecting the input to the desired output, is learned from data [108]. An end-to-end learning approach greatly reduces the need for prior knowledge about the problem, and minimises the required engineering effort; only the tuning of the model hyperparameters requires some expertise, but even that process can be automated [13]. Learning features can result in better performance than engineering them, because they are automatically tailored to the task at hand. Furthermore, training the entire processing architecture can lead to new insights about what kind of information is salient for a given task [67, 56].

Convolutional neural networks (CNNs) [83] in particular lend themselves well to this setting, because they consist of many layers of processing that are all learned using the same objective function, which is propagated through the network. CNNs have been used for image classification [73, 84], speech recognition [85], epileptic seizure detection [104], and many other applications. In music information retrieval, they have been used for onset detection

[77], genre classification [85, 88, 33], artist recognition [85, 33], instrument classification [66] and content-based recommendation [149].

From a biological perspective, computing mid-level representations from audio signals before applying learning algorithms to them makes sense: the human brain also perceives sound in the form of a mid-level representation. This results from sound passing through the cochlea, which is in essence a mechanical filter bank. Different parts of the cochlea respond to different frequencies. Nevertheless, it is still interesting to attempt to learn this part of the processing pipeline as well. Some information in raw audio signals is discarded by commonly used mid-level representations (most notably the phase). Learning from raw audio signals ensures that no potentially relevant information is discarded, and that the learned representations are tuned to the task at hand.

5.4 Experiments and results

To compare end-to-end learning with the traditional MIR approach of using a mid-level representation of the audio signals, I trained deep CNNs to perform automatic tagging on the Magnatagatune dataset [81]. The dataset contains 25863 29-second audio clips with a sample rate of 16 kHz, taken from songs by 230 artists, annotated with 188 tags. It comes in 16 parts, of which I used the first 12 for training, the 13th for validation and the remaining 3 for testing. I only used the 50 most frequent tags to ensure that enough training data was available for all of them.

5.4.1 Experimental setup

The CNN architecture that I used as a basis for all our experiments is visualised in Figure 5.1. It consists of 6 layers in total: two convolutional layers with 32 filters of length 8, alternating with max-pooling layers with pooling size 4, and two dense layers with 100 and 50 units respectively. I used rectified linear units [110] in all layers except for the top layer, where I used sigmoidal units. I extended this architecture for each the experiments described in the following subsections. Note that all convolutions and pooling operations are one-dimensional, i.e. only along the axis representing time. Although it is possible to convolve and pool in the frequency direction in the case of spectrogram input [1], I did not investigate this approach here to ensure a fair comparison, as it is not possible with raw audio input.

I trained the network using minibatch gradient descent, with minibatches of 10 examples. I used windows of about 3 seconds of audio as input. To

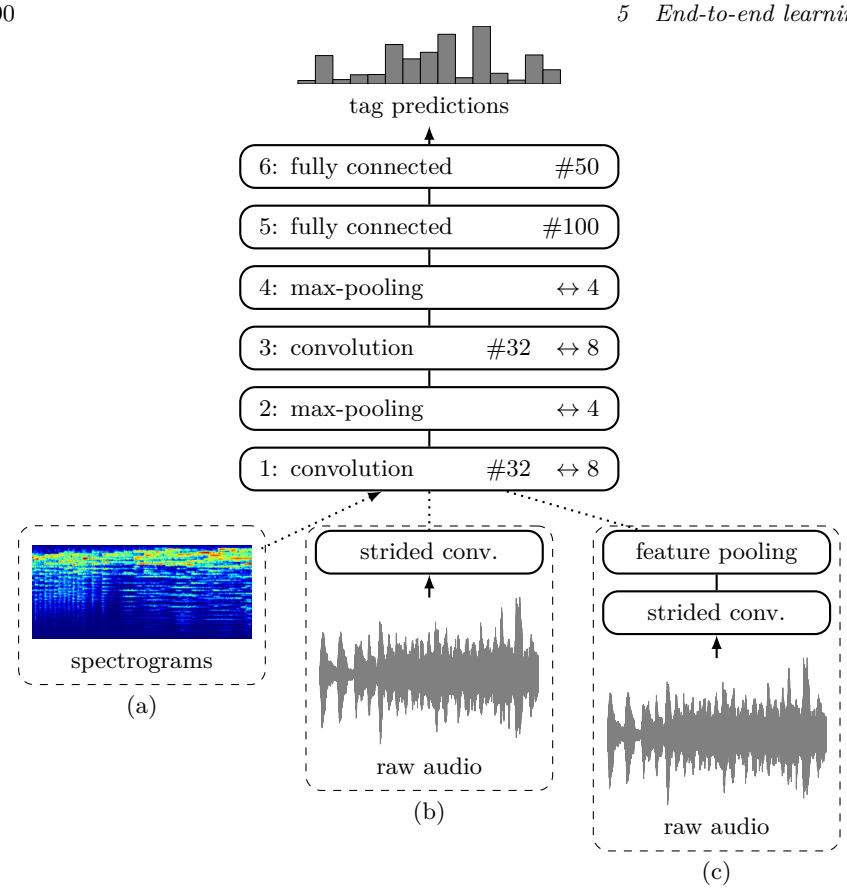


Figure 5.1: The convolutional neural network architecture I used for the experiments. The filter sizes and pooling sizes (↔) and numbers of units (#) are indicated. I consider three possible approaches: (a) spectrograms as input, (b) raw audio as input by adding an additional strided convolutional layer, and (c) raw audio with feature pooling.

compute tag predictions for a clip, I averaged the predictions over consecutive windows. I evaluated the use of dropout regularization [57] in the fully connected layers, but this did not affect performance significantly. To evaluate the predictions, I computed the area under the ROC curve (AUC) for each tag and computed the average across all 50 tags. For each experiment, I performed roughly 5 million parameter updates and validated the model at regular intervals. I report results on the test set for the parameters that achieved the best validation score. I used the Theano library to enable GPU acceleration [14].

length	**stride**	**AUC** (spectrograms)	**AUC** (raw audio)
1024	1024	0.8690	0.8366
1024	512	0.8726	0.8365
512	512	0.8793	0.8386
512	256	0.8793	0.8408
256	256	0.8815	0.8487

Table 5.1: Results for the tag prediction task on Magnatagatune, with convolutional neural networks using spectrograms and raw audio, for different combinations of filter lengths and strides. AUCs on the test set are reported. 1024 samples correspond to 64 ms at a sample rate of 16 kHz.

5.4.2 Spectrograms versus raw audio

To assess whether the task of tag prediction can be solved with a CNN using only raw audio, I compared two approaches:

- **spectrograms**: I extracted mel-spectrograms with 128 components, and performed dynamic range compression by applying the element-wise function $f(x) = \log(1 + C \cdot x)$, where C is a constant controlling the amount of compression [106], which I set to $10,000$. This representation was used as input to the network, as shown in Figure 5.1a. I tried a number of different window lengths and strides (hop sizes), which are listed in Table 5.1.

- **raw audio**: I extended the basic architecture described in the previous subsection by adding an additional convolutional layer at the input side, and used raw audio as input, as shown in Figure 5.1b. The additional layer performs a *strided convolution*, i.e. a convolution with a stride larger than one, because it would be computationally infeasible to compute full convolutions on raw audio signals. For this layer, I tried the same lengths and strides as for the spectrogram extraction. The raw audio signals were not preprocessed in any way.

The results for these experiments are listed in Table 5.1. The spectrogram approach consistently outperforms using raw audio, and a smaller window length consistently improves results. Presumably the increased time granularity is useful for this task. Reducing the stride to half the window length does not result in any significant improvement.

To see whether the network using raw audio as input is able to learn frequency-selective filters in the lowest layer, I computed their squared magnitude spectra. I have visualised the spectra of the filters ordered according

to the dominant frequency (low to high) in Figure 5.2. From this visualisation, it is clear that most features are indeed frequency-selective, and they cover the lower half of the spectrum. This is to be expected, as the harmonic content of music tends to be mostly in the lower end of the spectrum. It is especially worth noting that the filters seem to linearly span the frequency range up to about 1000 Hz, whereas filters that are selective for higher frequencies are more spread out. This is reminiscent of the mel scale. Some of the learned filters are shown in Figure 5.3, ordered by dominant frequency. They are quite noisy, but the dominant frequency is visible for most.

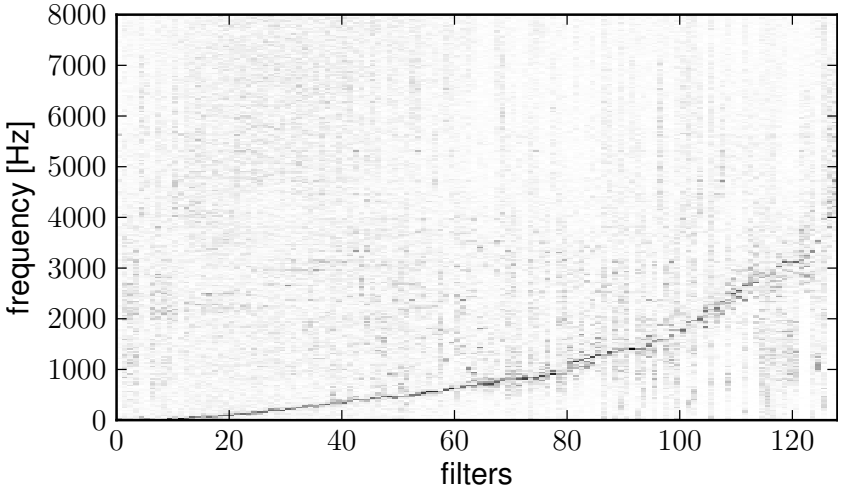


Figure 5.2: Normalised magnitude spectra of the filters learned in the lowest layer of a convolutional neural network that processes raw audio signals, ordered according to the dominant frequency (from low to high). Each vertical line in the graph represents the frequency spectrum of a different filter. The filters have a length and stride of 512 samples.

Despite the fact that the training procedure is able to discover a frequency decomposition from raw audio examples, there is a significant performance gap between both approaches. This may be because the network architecture for raw audio input is not sufficiently expressive to perform an operation similar to spectrogram extraction. In the following experiments, I have attempted to incorporate some aspects of spectrograms into the network architecture, to assess whether they are important to achieve good performance.

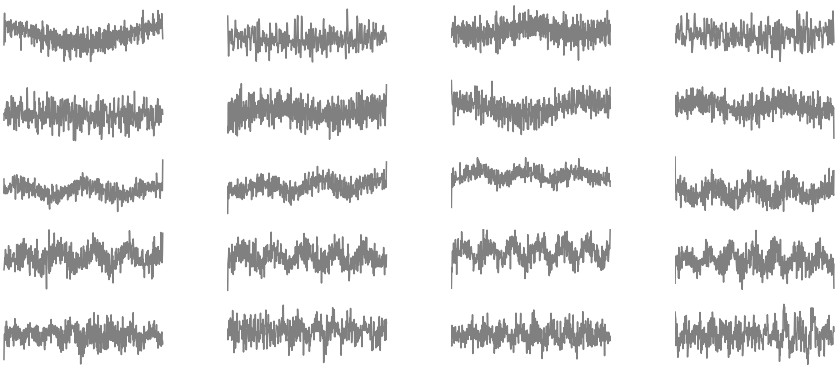


Figure 5.3: A subset of filters learned in the lowest layer of a convolutional neural network that processes raw audio signals, ordered by dominant frequency. The filters have a length and stride of 512 samples.

5.4.3 Dynamic range compression

The mel-spectrograms were compressed using the nonlinear function described in Section 5.4.2. This type of nonlinearity is probably difficult to learn using a network consisting of rectified linear units. I replaced the rectification nonlinearity in the strided convolutional layer with a compression function. The results are shown in Table 5.2. Unfortunately, this does not have the desired effect, and in fact it severely degrades performance. The logarithmic nonlinearity seems to impede the optimization process.

5.4.4 Invariance

Spectrograms exhibit various types of invariance, which are likely to be useful for the task of tag prediction: they are phase-invariant, and translation-

nonlinearity	$f(x)$	**AUC**
rectified linear	$\max(0, x)$	0.8366
logarithmic	$\log(1 + C \cdot x^2)$	0.7508
logarithmic	$\log(1 + C \cdot \lvert x \rvert)$	0.7487

Table 5.2: Results for the tag prediction task on Magnatagatune, with convolutional neural networks using raw audio as input, for different types of nonlinearities used in the strided convolutional layer. AUCs on the test set are reported. A filter length and stride of 1024 were used for all experiments.

pooling	pool size	AUC
no pooling	1	0.8366
L2 pooling	2	0.8387
L2 pooling	4	0.8387
max-pooling	2	0.8183
max-pooling	4	0.8280

Table 5.3: Results for the tag prediction task on Magnatagatune, with convolutional neural networks using raw audio as input, for different types of feature pooling after the strided convolutional layer. AUCs on the test set are reported. A filter length and stride of 1024 were used for all experiments.

invariant to a limited extent as well. Mel-scaled spectrograms are also somewhat frequency-invariant, particularly for higher frequencies. Automatically discovering these invariances from data may be quite challenging. To facilitate this process, we can further modify the network architecture to pool across groups of filters, as shown in Figure 5.1c. Hyvärinen and Hoyer [68] showed that summing the squares of the activations of a set of linear filters (L2 pooling) allows for learning phase- and translation-invariant features. More recently, units computing the maximal activation across a set of linear filters (*maxout* units) have been used to achieve state of the art image classification performance on several benchmark datasets [49]. I have evaluated both approaches.

The results are shown in Table 5.3. Although L2 pooling does not seem to perform significantly better than no pooling, and max-pooling performs worse, it should be noted that combining linear filters by pooling reduces the effective number of features computed in the strided convolutional layer. This in turn reduces the number of parameters in the next convolutional layer, leading to a network with fewer trainable parameters. Some filters learned by the network with L2 pooling and a pool size of 4 are shown in Figure 5.4. As expected, most of the pools consist of filters that are translated or phase-shifted versions of each other.

5.5 Conclusion

In this chapter, I have investigated whether end-to-end learning for music audio is feasible using convolutional neural networks to solve an automatic tagging task. Although the performance level of a spectrogram-based approach was not reached, I have shown that the networks are able to learn

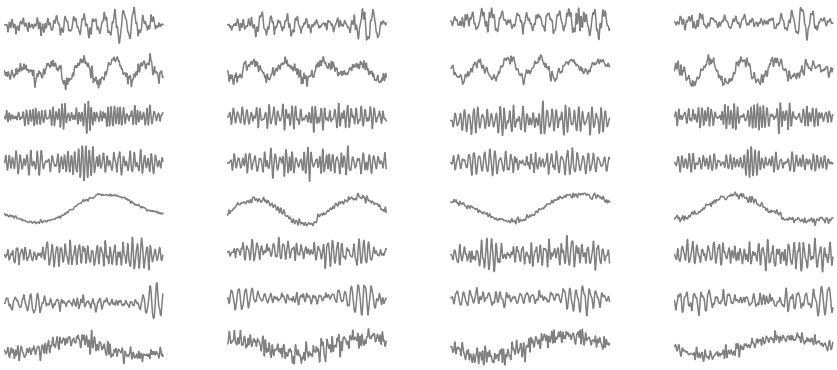


Figure 5.4: A subset of filters learned in a convolutional neural network with a feature pooling layer (L2 pooling with pools of 4 filters). The filters have a length and stride of 1024 samples. Each row represents a filter group. The filters were low-pass filtered to remove noise and make the dominant frequencies stand out.

useful features from raw audio: they are able to autonomously discover frequency decompositions, and when a feature pooling layer is incorporated, they discover phase- and translation-invariant features as well.

Given larger datasets and more complex tasks, which provide opportunities to train much larger and deeper networks, I believe the performance of a network trained on raw audio signals should eventually be able to match or even exceed that of a network trained on spectrogram inputs. Large networks are expressive enough to reproduce (or indeed improve on) the spectrogram extraction operation in their lower layers, and if there is enough available data they should be able to learn this function without overfitting. This could also be verified by initializing the parameters of the lower layers in such a way that they already perform this operation without any training, and subsequently finetuning them further through backpropagation.

Within the deep learning community, there has been a noticeable increased interest in generative models in the past two years. Although most research is focused on the generation of images or discrete sequences, generating audio signals is also an interesting application. In this line of research it would also be useful to eliminate mid-level representations and let the models generate raw audio signals directly, because inverting mid-level representations can be cumbersome. Future work on using neural networks with raw audio signals is likely to be situated chiefly in this domain.

6 Conclusion and perspectives

In this chapter, I will provide a summary of research conclusions. I will also review some potential directions for future work. In a broad sense, I will discuss my view of the future of deep learning and feature learning for MIR applications. In a more narrow sense I will also provide some potential follow-up work for the material covered in this thesis.

6.1 Summary

The ongoing digitization of the music industry, which started around the turn of the century with the advent of online music stores and later streaming services, has resulted in an increased interest in music information retrieval (MIR) in recent years, and in content-based MIR in particular. It has created a need to efficiently organize and categorize large collections of music, and to make them easily browseable and searchable.

In this thesis, I have explored how deep learning and feature learning techniques can be used for music classification, tagging and recommendation, task that are of considerable interest to music retailers, artists and listeners alike. I have developed content-based MIR approaches for these tasks that exploit the inherent hierarchical structure present in music. Deep neural networks allow for this hierarchy to be learnt from data while requiring only a minimal amount of domain knowledge.

In chapter 2 I developed a model for audio-based music classification using a pre-trained convolutional neural network. I made use of the Million Song Dataset (MSD), a large collection of metadata and precomputed audio features for one million songs. I demonstrated how convolutional neural networks are able to harness large amounts of unlabeled data through unsupervised pre-training, which is especially useful when labeled training data

is scarce. Most work on audio-based music classification thus far has been limited by the lack of industry-scale music datasets to develop and test new methods on. This work was some of the first in this domain taking advantage of the availability of the MSD, a dataset that is orders of magnitude larger than was typically the case in MIR research until then.

The work was limited in other ways, however: only a limited number of precomputed audio features were provided with the MSD, and there was no way to obtain raw audio signals, or even excerpts of them. This means that, while the proposed approach was based on feature learning, only higher-level features could be learned. The lower-level features were fixed by the authors of the dataset. Furthermore, some of the metadata provided with the dataset is not necessarily suitable for evaluating classification tasks. For example, the musical key information I used was computed using an algorithm and not manually annotated, reducing its usefulness.

The work should also be viewed in context - it predates a number of high-impact changes in how deep learning is typically practiced, such as the use of piecewise linear activation functions and dropout regularization. Nevertheless, the conclusion that unlabeled data can be valuable for MIR classification tasks still stands today, and will only become more relevant as music collections grow larger and manual annotation becomes increasingly unwieldy.

Subsequently I investigated an alternative approach to build a feature hierarchy for music classification and automatic tagging in chapter 3. Rather than having the levels of the hierarchy build upon each other, I proposed a purely temporal hierarchy where higher-level features simply encompass coarser timescales, but all features are learned directly from (temporally downsampled versions of) mid-level representations. While this does not qualify as deep learning because only one level of features is learned, this hierarchy also allows for patterns to be captured at different levels of abstraction, because in music higher abstraction levels tend to manifest themselves on coarser timescales.

I used an unsupervised feature learning technique based on the spherical K-means algorithm. This has many advantages: the training procedure is very fast and the algorithm is simple to understand and implement. It also turns out that the resulting feature representations are quite effective for the classification and tagging tasks I investigated. A downside is that higher level features necessarily have a much lower resolution. So while this method is able to capture high-level features, it cannot capture them in as much detail. As a result it is only useful for tasks where detailed temporal structure on longer timescales is not as important. With a more traditional 'deep' feature hierarchy, where higher level features build upon lower level features, this problem does not occur.

In chapter 4, I describe a new approach to content-based music recommendation which my colleague Aäron van den Oord and I developed together. We obtain latent factor vector representations of songs from historical listening data using collaborative filtering techniques and then build a convolutional neural network to map audio signals into this vector space, reducing content-based music recommendation to a regression problem in the process. Through both qualitative and quantitative evaluation, we showed that this approach yields sensible recommendations for songs for which no listening history is available, which makes it suitable for cold-start recommendations.

Despite these promising results, audio-based music recommendation is only in the early stages of adoption. Commercial services that provide music recommendations still rely mostly on a traditional collaborative filtering approach, often combined with human curation. For content-based recommendation to become practically usable and economically viable, more research is needed into how it can be combined with the aforementioned traditional approach so as to profit from the advantages of both. Although both collaborative filtering and content-based recommendation have been shown to be very effective for specific use cases, hybrid systems rarely show significant overall improvements.

Furthermore, the market for content-based recommenders is relatively small: most listeners are not particularly interested in receiving recommendations that are similar to their known preferences in terms of sound - they may prioritize other attributes (such as novelty, popularity and other social aspects), or they may not be interested in recommendations at all. Those that are tend to be music fans who actively seek out new artists and albums to discover.

This implies that content-based recommendation needs to reach a very high standard of quality before it is worth investing in: it can improve the user experience, but only for the subset of the user base that fits this particular profile, and only consistent high quality recommendations will appease this group and create a sense of trust. Services like SoundCloud and Bandcamp tend to cater specifically to this type of listener, so they might benefit more from content-based recommendations than mainstream streaming services and online music stores such as Spotify, Google Play Music, Pandora and iTunes.

The potential of our work was recognized by various music streaming and online music retailing companies, and lead to an internship for both Aäron and myself: he spent the summer with the Google Play Music team, and I spent three months at Spotify. Both of us had the opportunity to try out our method at an industrial scale.

In chapter 5, I investigated the necessity of mid-level time-frequency rep-

resentations in the context of automatic tagging of music with convolutional neural networks. One of the promises of deep learning is to make feature engineering and the incorporation of domain knowledge about the data and the problem largely obsolete - yet in the context of content-based MIR, the norm is to use mid-level representations of audio signals as input to learning algorithms, rather than the raw PCM-encoded audio signals themselves. I wanted to determine whether this is strictly necessary, and if it is possible to do away with this form of domain knowledge altogether.

Although convolutional neural networks trained on raw audio signals were not able to achieve the same level of automatic tagging performance as those trained on spectrogram representations, these networks were able to autonomously discover phase- and translation-invariant frequency decompositions that resemble the mel-scale, purely by supervised training. With larger datasets and further tuning of the network architecture, I believe that networks trained on raw audio signals should be able to match, and possibly even exceed the performance of those trained on spectrograms.

In summary, my exploration of solutions for MIR problems based on deep learning and feature learning has demonstrated that these techniques will potentially have a huge impact on the field – as has already been the case for various other fields related to signal processing such as computer vision, speech recognition and natural language processing.

6.2 Perspectives

In the future, MIR applications will only gain importance as the music industry continues to move to digital distribution. Music streaming is growing rapidly, with Spotify doubling its number of paying subscribers from 10 million to 20 million over the past year[1] and new competitors such as Google, Apple and Tidal entering the streaming market. In this context, I would like to propose a few immediate extensions of the work described in this thesis, as well as some longer-term ideas and research questions that may be of interest.

6.2.1 Pre-trained convolutional neural networks for music classification

As discussed in the previous section, my work on music classification using a pre-trained convolutional neural network is quite dated by now. The results

[1] https://news.spotify.com/uk/2015/06/10/20-million-reasons-to-say-thanks/

could be improved considerably by modernising the network architecture: using different nonlinearities, making the network deeper, and using better regularization and learning algorithms. Furthermore, the work would benefit from a more end-to-end approach: rather than having the network learn high-level features based on pre-extracted handcrafted features, the entire feature hierarchy could be learned from raw audio signals or a mid-level representation such as spectrograms.

6.2.2 Multiscale representations

My research on multiscale representations could be extended to be more suitable for tasks that rely on precise time information, by refraining from aggregating the features across time using a pooling strategy (i.e. the bag-of-words approach) and using smarter approaches to incorporate long-range temporal structure. The recent surge of interest in recurrent neural networks (RNNs), and more specifically long short-term memory networks (LSTM networks) has resulted in the development of a versatile framework for the processing of time series, which could be of use for e.g. cover song detection, score following or the identification of individual songs.

The multiscale representations based on spherical K-means feature learning were only tested for tagging and classification tasks. Their merits for other tasks, such as audio-based music recommendation, were never established. It would be interesting to determine how well they fare against convolutional neural network-based approaches in this context – in other words, to establish the importance of hierarchies based on depth, rather than on multiple timescales for this particular task. Multiscale representations have the advantage that each level in the hierarchy can be learned independently, and the learning procedures tend to be less computationally intensive. These traits could be beneficial for large-scale music recommendation.

The proposed approach for obtaining multiscale representations resulted in quite some redundancy between different levels. I have already investigated the use of Laplacian pyramids to avoid this, but it should be possible to remove more of this redundancy and reduce the size of the representations without losing any information in the process.

My work on multiscale representations relied solely on unsupervised learning, whereas the deep hierarchical representations described throughout this thesis were learned in a supervised or semi-supervised fashion (typically using convolutional neural networks). When available, supervision is valuable because it allows for feature representations to be adapted and specialised for the task at hand. Indeed, many state-of-the-art results in deep learning of late have been obtained using supervised learning exclusively.

Therefore it would be interesting to explore supervised learning in combination with multiscale representations, as well as unsupervised approaches for learning deep hierarchical representations. This would allow for disentangling the choice between supervised and unsupervised approaches on the one hand, and the choice between deep and multiscale representations on the other hand. Both choices could then be studied individually.

Finally, a feature hierarchy need not be either deep or multiscale – it could also be both. I already briefly explored this in Section 3.5.6, but a more in-depth treatment could provide insight into the relation between abstraction levels and musical timescales.

6.2.3 Content-based music recommendation

As stated before, content-based music recommendation is not widely in use at the time of writing. Further research is required before we will be able to build systems that produce high quality recommendations that meet the required standard of quality for economic viability.

Better integration with traditional techniques such as collaborative filtering will be instrumental, seeing as they are largely complementary. While collaborative filtering provides adequate recommendations in many situations, content-based approaches could be used to filter outliers, or to provide good recommendations in contexts where historical preference information is not available. Furthermore, I believe there should be a focus on producing consistent high quality recommendations and avoiding missteps: gaining the users' trust is very important for a recommender system to be successful.

Another promising avenue of future research concerns the interpretability of the recommendations. Content-based approaches have the potential to make the underlying motivation for a recommendation much more explicit, by having the recommender point out specific passages or even acoustic characteristics that led it to make a certain recommendation, as well as providing comparisons to other songs. For example, the system could provide the explanation that a recommended song contains a guitar solo that is similar in style to those in two other songs that the listener has previously enjoyed. It could then provide said listener with the option to play back the relevant parts of those two songs, in order to verify the similarity for themselves. Such explanations should increase trust and engagement and make the recommender more useful.

Note that, although neural networks have long had the undeserved reputation of being black boxes that are unable to provide these explanations, a plethora of recent research has demonstrated that features learned by these models are often very interpretable, and some new techniques to obtain such interpretations have been developed in the last two years [159, 133, 95, 2].

This work has focussed mostly on the interpretation of image features, but almost all of it should be equally applicable to audio signals with minor modifications.

Once the aspects and relations that motivate recommendations have been made explicit, we can also exploit them to make the recommendations context-dependent, or even tunable: we can give the listener the ability to (de-)emphasize certain aspects of the music that they feel are (less) important, allowing them to guide the recommendation process. This will likely be appreciated in particular by those listeners that would benefit the most from audio-based recommendations in the first place.

6.2.4 End-to-end feature learning

Although my experiments showed that training convolutional neural networks on raw audio signals rather than mid-level representations results in slightly worse performance, these results were very promising considering the relatively small scale of the automatic tagging task. By scaling up the complexity of the task, as well as drastically increasing the amount of available training data, it should be possible to match and maybe even exceed the performance of the approach based on mid-level representations using only raw audio signals.

Reducing the amount of prior knowledge required to achieve good results is valuable: if enough data is available, better representations might be learnt than those we can handcraft. In that sense, it is interesting that the networks I trained on raw audio were able to rediscover frequency decompositions and the mel-scale. As an intermediate step, it would be interesting to initialize the lower layers of a network that takes raw audio as input in such a way that it performs spectrogram extraction. This operation can then be finetuned during training.

In the long term, tools and techniques for processing raw audio with neural networks will be important for building generative models of audio signals. Although generative models can be built at the spectrogram level, resynthesizing spectrograms is sometimes challenging because phase information is usually discarded (and it is much harder to model).

6.2.5 Long-term perspectives

6.2.5.1 Modeling long-range structure

A limitation of many MIR systems (including those proposed in this thesis) is that they operate on fragments of songs, not on entire songs. This is often done for convenience or to reduce computational requirements. It hinges on

the assumption that a fragment of a song is representative for the song as a whole[2].

The high-level structure of music that manifests itself on longer timescales is not modeled by these systems. This is in part because of diminishing returns: modeling this structure is difficult and computationally intensive, and for many common MIR tasks the resulting performance improvement would likely be marginal. Another reason is that it often requires a different modeling approach: both traditional methods and convolutional neural networks are not very suitable.

Building integrated models that are able to model both local and global structure efficiently is an important challenge that has not yet been met. I believe that recurrent neural networks (RNNs) and especially long short-term memory networks (LSTMs), whose popularity has steadily risen over the last few years and which are now in common use for the classification and generation of sequences, hold a lot of promise for MIR as well. A combination of convolutional layers for modeling local temporal structure, as well as recurrence for modeling long-range structure could prove very effective.

6.2.5.2 Industrial-scale training and evaluation

Another important avenue for future work is the application of the proposed methods to large-scale datasets with better annotations. Unfortunately this is not possible today, because such datasets are simply unavailable. The Million Song Dataset, the largest dataset used in this work, is at least an order of magnitude smaller than the catalogs of music streaming services such as Spotify and Apple Music.

To ensure the relevance of future research into audio-based music classification and recommendation, access to larger datasets will be a necessity. Furthermore, it has already been demonstrated in other fields (most notably computer vision) that deep learning approaches scale very well with dataset size. In addition to improved performance, larger datasets will also allow for the further reduction of prior knowledge encoded in the models. In other words, the models will be able to learn more with more data.

6.2.5.3 Generative models for music audio

The work in this thesis focuses on extracting information from music audio, but another interesting research question is how to build models that are able

[2]This assumption is not always valid. An interesting counter-example is the song 'Bohemian Rhapsody' by Queen.

to generate plausible-sounding musical audio signals given some kind of high-level representation of the desired characteristics. Deep learning holds a lot of promise here as well. Research into generative models of images has made great strides in recent years, thanks to the development of better models such as deep Boltzmann machines (DBMs) [120], variational autoencoders (VAEs) [72] and generative adversarial networks (GANs) [48], all of which should be applicable to music audio as well. As stated before, I believe it will be especially important to develop good tools to work with raw audio signals in combination with these models, as mid-level representations often discard phase information and are difficult to invert without artifacts.

A Galaxy morphology classification

During my time as a PhD student I participated in a number of data science competitions. One of these competitions was the *Galaxy Challenge*[1]. It was hosted online on the Kaggle platform[2] and ran from December 20th, 2013 to April 4th, 2014. The goal was to build a system for automated morphological classification of galaxy images. There were 326 participants, including both teams and individuals. I finished in the first place using an approach based on convolutional neural networks.

Although this work falls outside of the scope of this thesis seeing as it does not concern a music information retrieval application, I wanted to include it because it demonstrates the versatility and adaptability of deep learning and feature learning tecnhiques: the entire pipeline is learnt from data, so these techniques can readily adapt to various different types of input data and can be applied to new problems with only a minimum of domain knowledge. Another motivation for including a description of this work in my thesis is that I spent a substantial amount of time on this work, and leaving it out entirely would seem inappropriate.

This appendix is a lightly edited version of a paper about the classification system I developed, which I wrote together with one of the competition organizers [34]. It was published in MNRAS, a prominent astronomy and astrophysics journal.

A.1 Introduction

Galaxies exhibit a wide variety of shapes, colours and sizes. These properties are indicative of their age, formation conditions, and interactions with

[1] https://www.kaggle.com/c/galaxy-zoo-the-galaxy-challenge
[2] https://www.kaggle.com/

other galaxies over the course of many Gyr. Studies of galaxy formation and evolution use morphology to probe the physical processes that give rise to them. In particular, large, all-sky surveys of galaxies are critical for disentangling the complicated relationships between parameters such as halo mass, metallicity, environment, age, and morphology; deeper surveys probe the changes in morphology starting at high redshifts and taking place over timescales of billions of years.

Such studies require both the observation of large numbers of galaxies and accurate classification of their morphologies. Large-scale surveys such as the Sloan Digital Sky Survey (SDSS)[3] have resulted in the availability of image data for millions of celestial objects. However, manually inspecting all these images to annotate them with morphological information is impractical for either individual astronomers or small teams.

Attempts to build automated classification systems for galaxy morphologies have historically had difficulties in reaching the levels of reliability required for scientific analysis [25]. The Galaxy Zoo project[4] was conceived to accelerate this task through the method of crowdsourcing. The original goal of the project was to obtain reliable morphological classifications for $\sim 900,000$ galaxies by allowing members of the public to contribute classifications via a web platform. The project was much more successful than anticipated, with the entire catalog being annotated within a timespan of several months (originally projected to take years). Since its original inception, several iterations of the project with different sets of images and more detailed classification taxonomies have followed.

Two recent sets of developments since the launch of Galaxy Zoo have made an automated approach more feasible: first, the large strides in the fields of image classification and computer vision in general, primarily through the use of *deep neural networks* [73, 118]. Although neural networks have existed for several decades [99, 43], they have recently returned to the forefront of machine learning research. A significant increase in available computing power, along with new techniques such as rectified linear units [110] and dropout regularization [57, 138], have made it possible to build more powerful neural network models.

Secondly, large sets of reliably annotated images of galaxies are now available as a consequence of the success of Galaxy Zoo. Such data can be used to train machine learning models and increase the accuracy of their morphological classifications. Deep neural networks in particular tend to scale very well as the number of available training examples increases. Nevertheless it is also possible to train deep neural networks on more modestly sized datasets using techniques such as regularization, data augmentation,

[3] `http://www.sdss.org/`
[4] `http://www.galaxyzoo.org/`

parameter sharing and model averaging, which we discuss in Section A.6.2 and following.

An automated approach is also becoming indispensable: modern telescopes continue to collect more and more images every day. Future telescopes will vastly increase the number of galaxy images that can be morphologically classified, including multi-wavelength imaging, deeper fields, synoptic observing, and true all-sky coverage. As a result, the crowdsourcing approach cannot be expected to scale indefinitely with the growing amount of data. Supplementing both expert and crowdsourced catalogues with automated classifications is a logical and necessary next step.

We propose a convolutional neural network model for galaxy morphology classification that is specifically tailored to the properties of images of galaxies. It efficiently exploits both translational and rotational symmetry in the images, and autonomously learns several levels of increasingly abstract representations of the images that are suitable for classification. The model was developed in the context of the *Galaxy Challenge*, an international competition to build the best model for automatic galaxy morphology classification based on a set of annotated images from the Galaxy Zoo 2 project. This model finished in first place out of 326 participants[5]. The model can efficiently and automatically annotate catalogs of images with morphology information, enabling quantitative studies of galaxy morphology on an unprecedented scale.

The rest of this appendix is structured as follows: we introduce the Galaxy Zoo project in Section A.2 and Section A.3 explains the set up of the Galaxy Challenge. We discuss related work in Section A.4. Our method to incorporate rotation invariance into convolutional neural networks is described in Section A.5. We provide a complete overview of our modelling approach in Section A.6 and report results in Section A.7. We analyse the model in Section A.8. Finally, we draw conclusions in Section A.9.

A.2 Galaxy Zoo

Galaxy Zoo is an online crowdsourcing project where users are asked to describe the morphology of galaxies based on colour images [91, 90]. Our model and analysis uses data from the Galaxy Zoo 2 iteration of the project, which uses colour images from the SDSS and a more detailed classification scheme than the original project [157]. Participants are asked various questions such as 'How rounded is the galaxy?' and 'Does it have a central bulge?', with the

[5]The model was independently developed by SD for the Kaggle competition. KWW co-designed and administered the competition, but shared no data or code with any participants until after the closing date.

users' answers determining which question will be asked next. The questions form a decision tree (Figure A.1) which is designed to encompass a range of common and more irregular morphologies. The classification scheme has 11 questions and 37 answers in total (Table A.1).

Because of the structure of the decision tree, each individual participant answered only a subset of the questions for each classification. When many participants have classified the same image, their answers are aggregated into a set of weighted vote fractions for the entire decision tree. These vote fractions are used to estimate confidence levels for each answer, and are indicative of the difficulty users experienced in classifying the image. More than half a million people have contributed classifications to Galaxy Zoo, with each image independently classified by 40 to 50 people[6].

Data from the Galaxy Zoo projects have already been used in a wide variety of studies on galaxy structure, formation, and evolution [134, 5, 123, 92, 30, 97, 98, 131, 103, 156]. Comparisons of Galaxy Zoo morphologies to smaller samples from both experts and automated classifications show high levels of agreement, testifying to the accuracy of the crowdsourced annotations [5, 157].

A.3 The Galaxy Challenge

Our model was developed in the context of the Galaxy Challenge, an online international competition organized by Galaxy Zoo, sponsored by Winton Capital, and hosted on the Kaggle platform for data prediction contests. It was held from December 20th, 2013 to April 4th, 2014. The goal of the competition was to build a model that could predict galaxy morphology from images like the ones that were used in the Galaxy Zoo 2 project.

Images of galaxies and morphological data for the competition were taken from the Galaxy Zoo 2 main spectroscopic sample. Galaxies were selected to cover the full observed range of morphology, colour, and size, since the goal was to develop a general algorithm that could be applied to many types of images in future surveys. The total number of images provided is limited both by the imaging depth of SDSS and the elimination of both uncertain and over-represented morphological categories as a function of colour (primarily red elliptical and blue spiral galaxies). This helped to ensure that colour is not used as a proxy for morphology, and that a high-performing model would be based purely on the images' structural parameters.

[6] Note that the vote fractions are post-processed to increase their reliability, for example by weighting users based on their consistency with the majority, and by compensating for classification bias induced by different image apparent magnitudes and sizes [157].

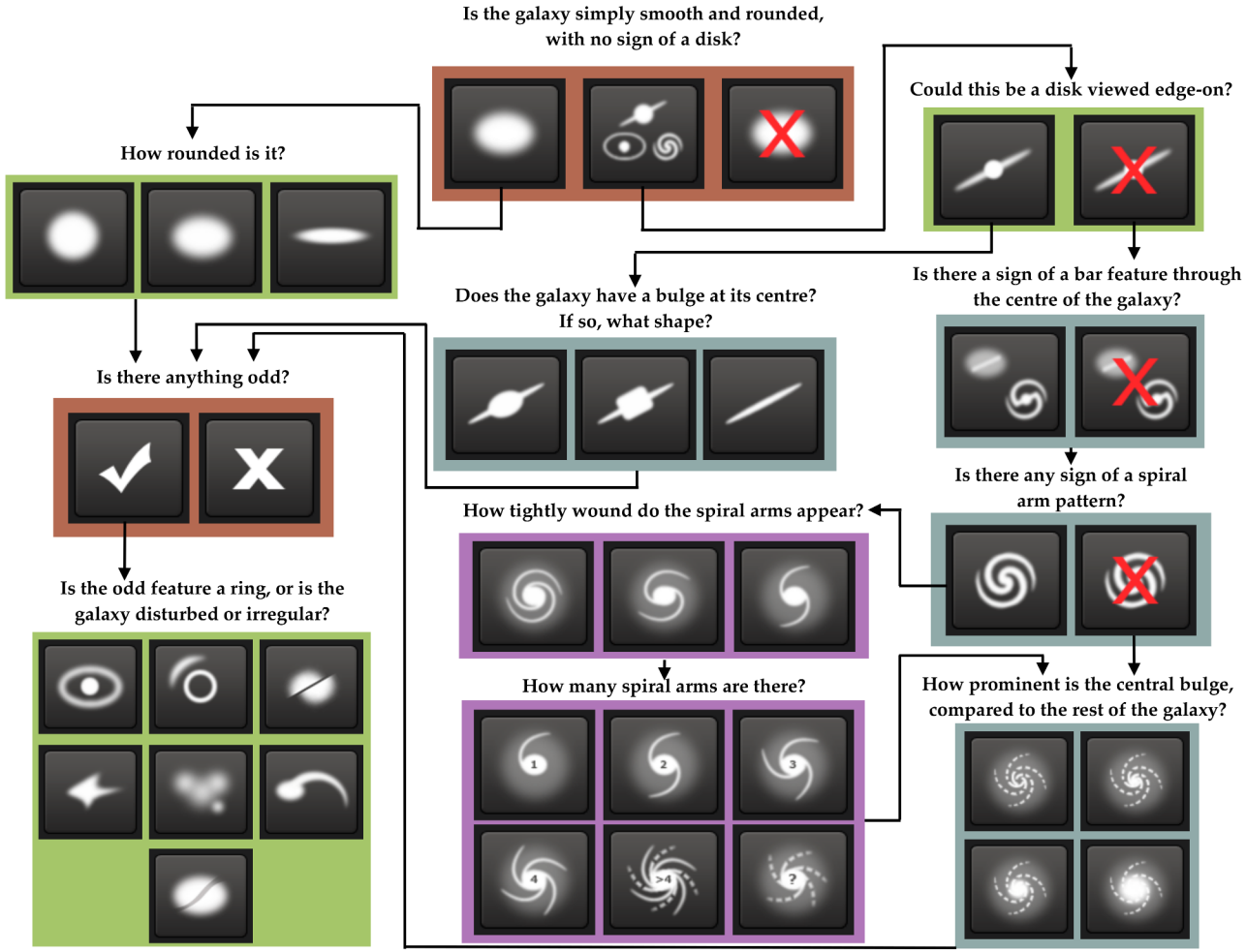


Figure A.1: The Galaxy Zoo 2 decision tree. Reproduced from Figure 1 in Willett et al. [157].

	question		answers	next
Q1	Is the galaxy simply smooth and rounded, with no sign of a disk?	A1.1	smooth	Q7
		A1.2	features or disk	Q2
		A1.3	star or artifact	end
Q2	Could this be a disk viewed edge-on?	A2.1	yes	Q9
		A2.2	no	Q3
Q3	Is there a sign of a bar feature through the center of the galaxy?	A3.1	yes	Q4
		A3.2	no	Q4
Q4	Is there any sign of a spiral arm pattern?	A4.1	yes	Q10
		A4.2	no	Q5
Q5	How prominent is the central bulge, compared with the rest of the galaxy?	A5.1	no bulge	Q6
		A5.2	just noticeable	Q6
		A5.3	obvious	Q6
		A5.4	dominant	Q6
Q6	Is there anything odd?	A6.1	yes	Q8
		A6.2	no	end
Q7	How rounded is it?	A7.1	completely round	Q6
		A7.2	in between	Q6
		A7.3	cigar-shaped	Q6
Q8	Is the odd feature a ring, or is the galaxy disturbed or irregular?	A8.1	ring	end
		A8.2	lens or arc	end
		A8.3	disturbed	end
		A8.4	irregular	end
		A8.5	other	end
		A8.6	merger	end
		A8.7	dust lane	end
Q9	Does the galaxy have a bulge at its center? If so, what shape?	A9.1	rounded	Q6
		A9.2	boxy	Q6
		A9.3	no bulge	Q6
Q10	How tightly wound do the spiral arms appear?	A10.1	tight	Q11
		A10.2	medium	Q11
		A10.3	loose	Q11
Q11	How many spiral arms are there?	A11.1	1	Q5
		A11.2	2	Q5
		A11.3	3	Q5
		A11.4	4	Q5
		A11.5	more than four	Q5
		A11.6	can't tell	Q5

Table A.1: All questions that can be asked about an image, with the corresponding answers that participants can choose from. Question 1 is the only one that is asked of every image. The final column indicates the next question to be asked when a particular answer is given. Reproduced from Table 2 in Willett et al. [157].

The final **training set** of data consisted of 61,578 JPEG colour images of galaxies, along with probabilities[7] for each of the 37 answers in the decision tree. An **evaluation set** of 79,975 images was also provided, but with no morphological data – the goal of the competition was to predict these values. Each image is 424 by 424 pixels in size. The morphological data provided was a modified version of the weighted vote fractions in the GZ2 catalog; these were transformed into "cumulative" probabilities that give higher weights to more fundamental morphological categories higher in the decision tree. Images were anonymized from their SDSS IDs, with any use of metadata (such as colour, size, position, or redshift) to train the algorithm explicitly forbidden by the competition guidelines.

Because the goal was to predict probabilities for each answer, as opposed to determining the most likely answer for each question in the decision tree, the models built by participants were actually solving a *regression* problem, and not a classification problem in the strictest sense. The predictive performance of a model was determined by computing the root-mean-square error (RMSE) between predictions on the evaluation set and the corresponding crowdsourced probabilities. Let p_k be the answer probabilities associated with an image ($k = 1 \dots 37$), and $\hat{p}_k$ the corresponding predictions. Then the RMSE $e(\hat{\mathbf{p}}, \mathbf{p})$ can be computed as follows:

$$e(\hat{\mathbf{p}}, \mathbf{p}) = \sqrt{\sum_{k=1}^{37} (\hat{p}_k - p_k)^2}. \tag{A.1}$$

This metric was chosen because it puts more emphasis on questions with higher answer probabilities, i.e. the topmost questions in the decision tree, which correspond to broader morphological categories.

The provided answer probabilities are derived from crowdsourced classifications, so they are somewhat noisy and biased in certain ways. As a result, the predictive models that were built exhibited some of the same biases. In other words, they are models of how *the crowd* would classify images of galaxies, which may not necessarily correspond to the "true" morphology. An example of such a discrepancy is discussed in Section A.8.

The models built by participants were evaluated as follows. The Kaggle platform automatically computed two scores based on a set of model predictions: a public score, computed on about 25% of the evaluation data, and a private score, computed on the other 75%. Public scores were immediately revealed during the course of the competition, but private scores were not revealed until after the competition had finished. The private score was used

[7] These are actually post-processed vote fractions obtained from the Galaxy Zoo participants' answers, but we treat them as probabilities in this paper.

to determine the final ranking. Because the participants did not know which evaluation images belonged to which set, they could not directly optimize the private score, but were instead encouraged to build a predictive model that generalizes well to new images.

A.4 Related work

Machine learning techniques, and artificial neural networks in particular, have been a popular tool in astronomy research for more than two decades. Neural networks were initially applied for star-galaxy discrimination [114, 15] and classification of galaxy spectra [39]. More recently they have also been used for photometric redshift estimation [38, 29].

Galaxy morphology classification is one of the most widespread applications of neural networks in astronomy. Most work in this domain proceeds by preprocessing the photometric data and then extracting a limited set of handcrafted features that are known to be discriminative, such as ellipticity, concentration, surface brightness, and radii and log-likelihood values measured from various types of radial profiles [141, 78, 109, 79, 4, 6]. Support vector machines (SVMs) have also been applied in this fashion [65].

Earlier work in this domain typically relied on much smaller datasets and used networks with very few trainable parameters (between 10^1 and 10^3). Modern network architectures are capable of handling at least $\sim 10^7$ parameters, allowing for more precise fits and a larger morphological classification space. More recent work has profited from the availability of larger training sets using data from surveys such as the SDSS [6, 65].

Another recent trend is the use of general purpose image features, instead of features that are specific to galaxies: the WND-CHARM feature set [115], originally designed for biological image analysis, has been applied to galaxy morphology classification in combination with nearest neighbour classifiers [128, 129, 76].

Other approaches to this problem attempt to forgo any form of handcrafted feature extraction by applying principal component analysis (PCA) to preprocessed images in combination with a neural network [31], or by applying kernel SVMs directly to raw pixel data [116].

Our approach differs from prior work in two main areas:

- Most prior work uses handcrafted features (e.g., WND-CHARM) that required expert knowledge and many hours of engineering to develop. We work directly with raw pixel data and our use of convolutional neural networks allows for a set of task-specific features to be learned from the data. The networks learn hierarchies of features, which allow

them to detect complex patterns in the images. Handcrafted features mostly rely on image statistics and very local pattern detectors, making it harder to recognize complex patterns. Furthermore, it is usually necessary to perform feature selection because the handcrafted representations are highly redundant and many features are irrelevant for the task at hand. Although many other participants in the Galaxy Challenge used convolutional neural networks, there is little discussion of this approach in the astronomical literature.

- Apart from the recent work of Kuminski et al. [76], whose algorithms are also trained on Galaxy Zoo data, most research has focused on classifying galaxies into a limited number of classes (typically between 2 and 5), or predicting scalar values that are indicative of galaxy morphology (e.g., Hubble T-types). Since the classifications made by Galaxy Zoo users are much more fine-grained, the task the networks must solve is more challenging. Since many outstanding astrophysical questions require more detailed morphological data (such as the number and arrangements of clumps into proto-galaxies, the relation between bar strength and central star formation, link between merging activity and active black holes, etc.), development of models that can handle these more difficult tasks is crucial.

Our method for classifying galaxy morphology exploits the rotational symmetry of galaxy images; however, there are other invariances and symmetries (besides translational) that may be exploited for convolutional neural networks. Bruna et al. [21] define convolution operations over arbitrary graphs, generalizing from the typical grid of pixels to other locally connected structures. Sifre and Mallat [130] extract representations that are invariant to affine transformations, based on scattering transforms. However, these representations are fixed (i.e., not learned from data), and not specifically tuned for the task at hand, unlike the representations learned by convolutional neural networks.

Mairal et al. [96] propose to train convolutional neural networks to approximate kernel feature maps, allowing for the desired invariance properties to be encoded in the choice of kernel, and subsequently learned. Gens and Domingos [44] propose deep symmetry networks, a generalization of convolutional neural networks with the ability to form feature maps over any symmetry group, rather than just the translation group. Our approach for exploiting rotational symmetry in the input images, described in Section A.5, is quite similar in spirit to this work. The major advantage to our implementation is a demonstrably effective result at a reasonable computational cost.

A.5 Exploiting rotational symmetry

Convolutional neural networks can be used when the input data exhibits some kind of topological structure, like the ordering of image pixels in a grid, or the temporal structure of an audio signal. The connectivity of the convolutional layers is restricted to exploit this structure and encode it in the network architecture. These restricted connectivity patterns drastically reduce the number of parameters required to model large images, by exploiting translational symmetry. However, there are many other types of symmetries that occur in images. For images of galaxies, rotating an image should not affect its morphological classification. This rotational symmetry can be exploited by applying the same set of feature detectors to various rotated versions of the input. This further increases parameter sharing, which has a positive effect on generalization performance.

Whereas convolutions provide an efficient way to exploit translational symmetry, applying the same filter to multiple rotated versions of the input requires explicitly instantiating these versions. Additionally, rotating an image by an angle that is not a multiple of 90° requires interpolation and results in an image whose edges are not aligned with the rows and columns of the pixel grid. These complications make exploiting rotational symmetry more challenging.

We note that the original Galaxy Zoo project experimented with crowdsourced classifications of galaxies in which the images were both vertically and diagonally mirrored. Land et al. [80] showed that the raw votes had an excess of 2.5% for S-wise (anticlockwise) spiral galaxies over Z-wise (clockwise) galaxies. Since this effect was seen in both the raw and mirrored images, it was interpreted as a bias due to preferences in the human brain, rather than as a true excess in the number of apparent S-wise spirals in the Universe. The existence of such a directional bias in the brain was demonstrated by Gori et al. [50]. The Galaxy Zoo 2 probabilities do not contain any structures related to handedness or rotation-variant quantities, and no rotational or translational biases have yet been discovered in the data. If such biases do exist, however, this would presumably reduce the predictive power of the model since the assumption of rotational invariance to the output probabilities would no longer apply.

Our approach for exploiting symmetry is visualized in Figure A.2. We compute rotated and flipped versions of the input images, which are referred to as *viewpoints*, and process each of these separately with the same convolutional network architecture, consisting of alternating convolutional layers and pooling layers. The output feature maps of this network for the different viewpoints are then concatenated, and one or more dense layers are stacked

on top. This arrangement allows the dense layers to aggregate high-level features extracted from different viewpoints.

In practice, we also crop the top left part of each viewpoint image both to reduce redundancy between the viewpoints and to reduce the size of the input images (and hence computation time). Images are cropped in such a way that each one contains the center of the galaxy, because this part of the image tends to be very informative. The practical implementation of viewpoint extraction is discussed in Section A.6.5, and the modified network architecture is described in Section A.6.6.

A.6 Approach

In this section, we describe our practical approach to developing and training a model for galaxy morphology prediction. We first discuss our experimental setup and the problem of overfitting, which was the main driver behind our design decisions. We then describe the successive steps in our processing pipeline to obtain a set of answer probabilities from an image. This pipeline consists of five steps (Figure A.3): input preprocessing, augmentation, viewpoint extraction, a convolutional neural network and model averaging. We also briefly discuss the practical implementation of the pipeline from a software perspective.

A.6.1 Experimental setup

As described in Section A.3, the provided dataset consists of a training set with 61,578 images with associated answer probabilities, and an evaluation set of 79,975 images. Feedback could be obtained during the competition by submitting predictions for the images in the evaluation set. During the competition, submitted predictions were scored by computing the RMSE on a subset of approximately 25% of the evaluation images. It was not revealed which images were part of this subset. The scores used to determine the final ranking were obtained by computing the RMSE on the remaining 75% of images. This arrangement is typical for competitions hosted on the Kaggle platform. We split off a further 10% of the training set images for real-time evaluation during model training, and trained our models only on the remaining 90%.

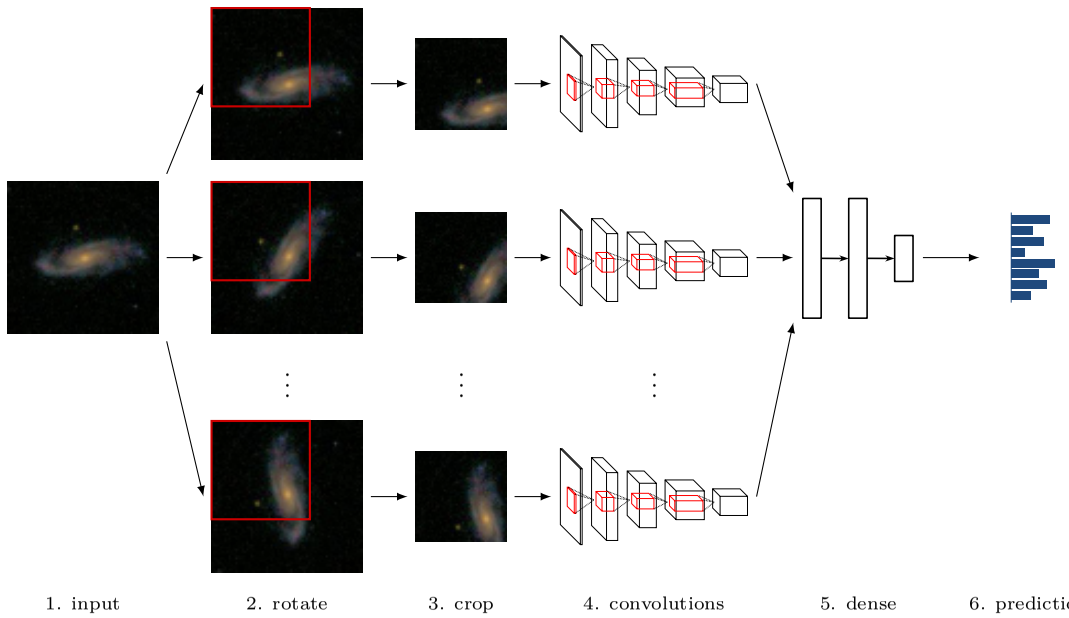


Figure A.2: Schematic overview of a neural network architecture for exploiting rotational symmetry. The input image (1) is first rotated to various angles and optionally flipped to yield different viewpoints (2), and the viewpoints are subsequently cropped to reduce redundancy (3). Each of the cropped viewpoints is processed by the same stack of convolutional layers and pooling layers (4), and their output representations are concatenated and processed by a stack of dense layers (5) to obtain predictions (6).

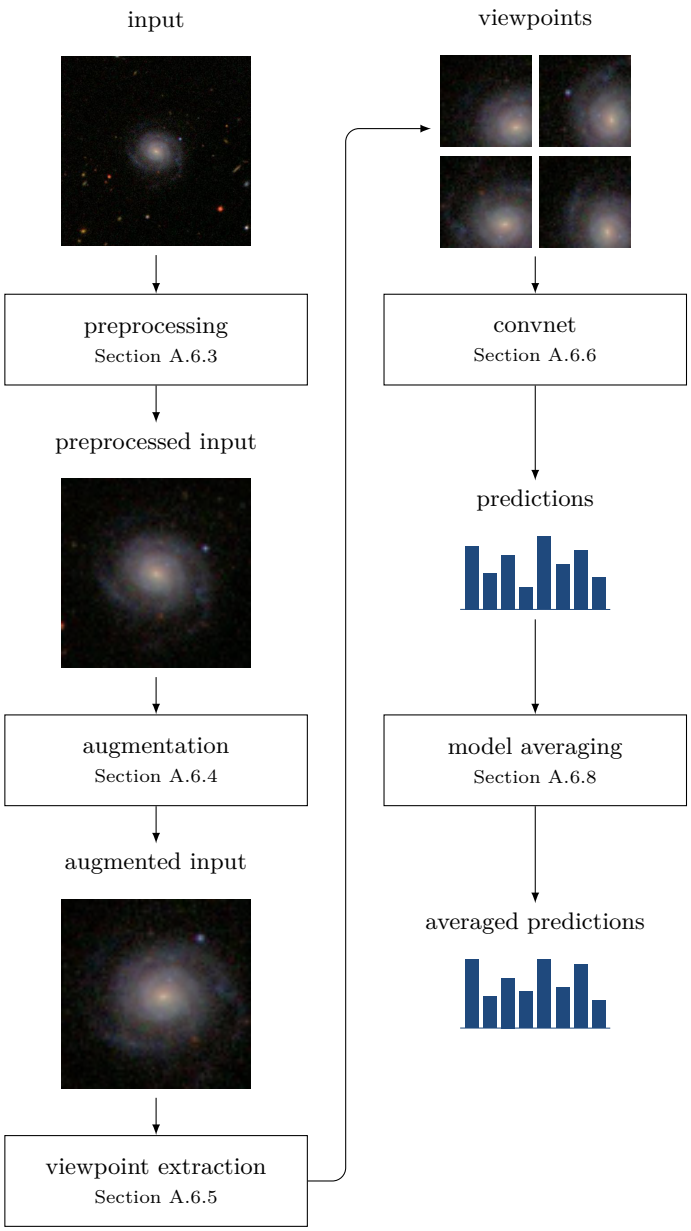


Figure A.3: Schematic overview of the processing pipeline.

A.6.2 Avoiding overfitting

Modern neural networks typically have a large number of learnable parameters – several million in the case of our model. This is in stark contrast with the limited size of the training set, which had only 5×10^4 images. As a result, there is a high risk of *overfitting*: a network will tend to memorize the training examples because it has enough capacity to do so, and will not generalize well to new data. We used several strategies to avoid overfitting:

- **data augmentation**: extending the training set by randomly perturbing images in a way that leaves their associated answer probabilities unchanged;
- **regularization**: penalizing model complexity through use of dropout [57];
- **parameter sharing**: reducing the number of model parameters by exploiting translational and rotational symmetry in the input images;
- **model averaging**: averaging the predictions of several models.

A.6.3 Preprocessing

Images are first cropped and rescaled to reduce the dimensionality of the input. It was useful to crop the images because the object of interest is in the middle of the image with a large amount of sky background, and typically fits within a square with a side of approximately half the image height. We then rescaled the images to speed up training, with little to no effect on predictive performance. Images were cropped from 424×424 pixels to 207×207, and then downscaled 3 times to 69×69 pixels.

For a small subset of the images, the cropping operation removed part of the object of interest, either because it had an unusually large angular size or because it was not perfectly centered. We looked into recentering and rescaling the images by detecting and measuring the objects in the images using *SExtractor* [16]. This allowed us to independently estimate both the position and Petrosian radii of the objects. This information is then used to center and rescale all images to standardize the sizes of the objects before further processing.

This normalization step had no significant effect on the predictive performance of our models. Nevertheless, we did train a few models using this approach, because even though they achieved the same performance in terms of RMSE compared to models trained without it, the models make different mistakes. This is useful in the context of model averaging (Section A.6.8),

where high variance among a set of comparably performing models is desirable [19].

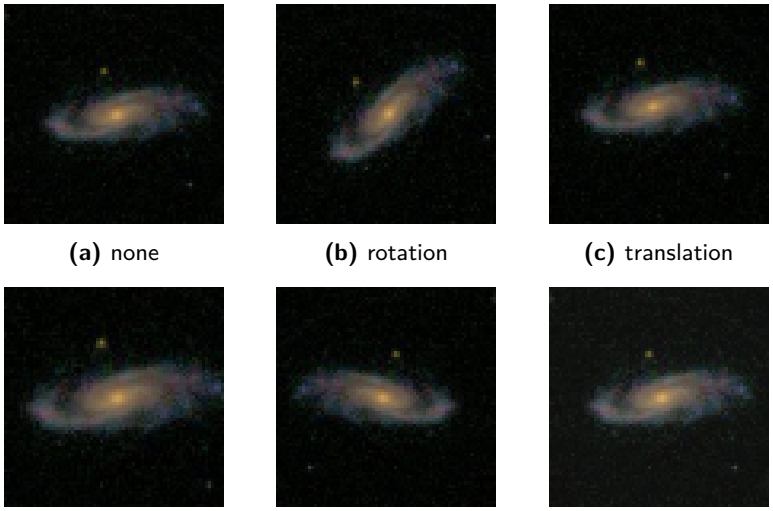
The images for the competition were provided in the same format that is used on the Galaxy Zoo website (424×424 JPEG colour images). We found that keeping the colour information (instead of converting the images to grayscale) improved the predictive performance considerably, despite the fact that the colours are artificial and intended for human eyes. These artificial colours are nevertheless correlated with morphology, and our models are able to exploit this correlation.

A.6.4 Data augmentation

Due to the limited size of the training set, performing data augmentation to artificially increase the number of training examples is instrumental. Each training example was randomly perturbed in five ways, which are shown in Figure A.4:

- **rotation**: random rotation with an angle sampled uniformly between 0° and 360°, to exploit rotational symmetry in the images.
- **translation**: random shift sampled uniformly between -4 and 4 pixels (relative to the original image size of 424 by 424) in the x and y direction. The size of the shift is limited to ensure that the object of interest remains in the center of the image.
- **scaling**: random rescaling with a scale factor sampled log-uniformly between 1.3^{-1} and 1.3.
- **flipping**: the image is flipped with a probability of 0.5.
- **brightness adjustment**: the colour of the image is adjusted as described by Krizhevsky et al. [73], with two differences: the first eigenvector has a much larger eigenvalue than the other two, so only this one is used, and the standard deviation for the scale factor is set to $\alpha = 0.5$. In practice, this amounts to a brightness adjustment.

The first four of these are affine transformations, which can be collapsed into a single transformation together with the one used for preprocessing. This means that the data augmentation step has no noticeable computational cost. To maximize the effect of data augmentation, we randomly perturbed the images on demand during training, so the models were never presented with the exact same training example more than once.



(a) none **(b)** rotation **(c)** translation

(d) scaling **(e)** flipping **(f)** brightness

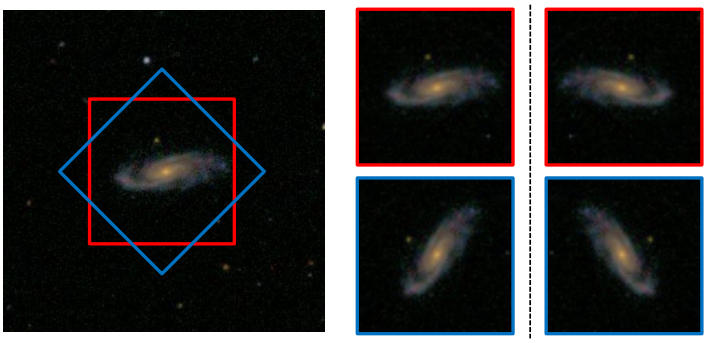
Figure A.4: The five types of random data augmentation used in this model. Note that the effect of translation and brightness adjustment is fairly subtle.

A.6.5 Viewpoint extraction

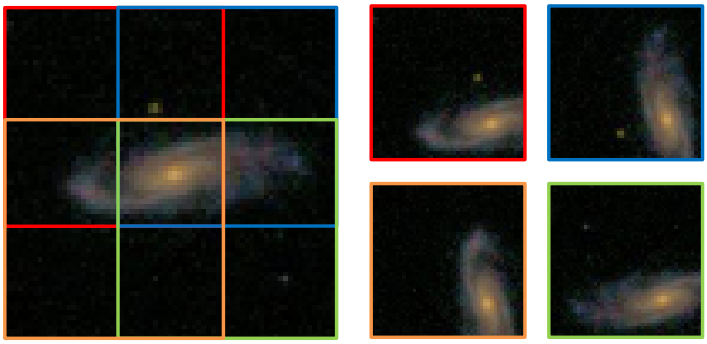
After preprocessing and augmentation, we extracted viewpoints by rotating, flipping and cropping the input images. We extracted 16 different viewpoints for each image: first, two square-shaped crops were extracted from an input image, one at 0° and one at 45°. Both were also flipped horizontally to obtain 4 crops in total. Each of these crops is 69×69 pixels in size. Then, four overlapping corner patches of 45×45 pixels were extracted from each crop, and rotated so that the center of the galaxy is in the bottom right corner of each patch. These 16 rotated patches constitute the viewpoints (Figure A.5).

This approach allowed us to obtain 16 different viewpoints with just two affine transformation operations, thus avoiding additional computation. All viewpoints can be obtained from the two original crops without interpolation[8]. This also means that image edges and padding have no effect on the input, and that the loss of image fidelity after preprocessing, augmentation and viewpoint extraction is minimal.

[8] In practice, extracting a viewpoint from these crops constitutes an array indexing operation.



(a) 4 crops from an image



(b) 4 viewpoints from each crop

Figure A.5: Obtaining 16 viewpoints from an input image. (a) First, two square-shaped crops are extracted from the image, one at 0° (red outline) and one at 45° (blue outline). Both are also flipped horizontally to obtain 4 crops in total. (b) Then, four overlapping corner patches are extracted from each crop, and they are rotated so that the galaxy center is in the bottom right corner of each patch. These 16 rotated patches constitute the viewpoints.

A.6.6 Network architecture

All viewpoints were presented to the network as 45 by 45 by 3 arrays of RGB values, scaled to the interval $[0, 1]$, and processed by the same convolutional architecture. The resulting feature maps were then concatenated and processed by a stack of three fully connected layers to map them to the 37 answer probabilities.

The architecture for the best performing network is visualized in Figure A.6. There are four convolutional layers, all with square filters, with filter sizes 6, 5, 3 and 3 respectively, and with untied biases (i.e. each spatial position in each feature map has a separate bias, see Section 1.3.4). The rectification non-linearity is applied after each layer [110]. 2 by 2 max-pooling follows the first, second and fourth convolutional layers. The concatenated feature maps from all viewpoints are processed by a stack of three fully connected layers, consisting of two maxout layers [49] with 2048 units with two linear filters each, and a linear layer that outputs 37 real numbers. Maxout layers were used instead of ReLU layers to reduce the number of connections to the next layer (and thus the number of parameters). We did not use maxout in the convolutional layers because it proved too computationally intensive.

We arrived at this particular architecture after a manual parameter search: more than 100 architectures were evaluated over the course of the competition, and this one was found to yield the best predictive performance. The network has roughly 42 million trainable parameters in total. Table A.2 lists the hyperparameter settings for the trainable layers.

The 37 values that the network produces for an input image are converted into a set of probabilities. First, the values are passed through a rectification non-linearity, and then normalized per question to obtain a valid categorical probability distribution for each question. Valid probability distributions could also be obtained by using a softmax function per question (as in a logistic regression model), instead of rectification followed by normalization. However, this decreases the overall performance since it is harder for the network to predict a probability of exactly 0 or 1.

The distributions still need to be rescaled, however; they give the probability of an answer conditional on its associated question being asked, but each user is only asked a subset of the questions. This implies that some questions have a lower probability of being asked, so the probabilities of the answers to these questions should be scaled down to obtain unconditional probabilities. In practice we scale them by the probabilities of the answers that preceded them in the decision tree (see Figure A.1).

This post-processing operation is incorporated into the network. Because

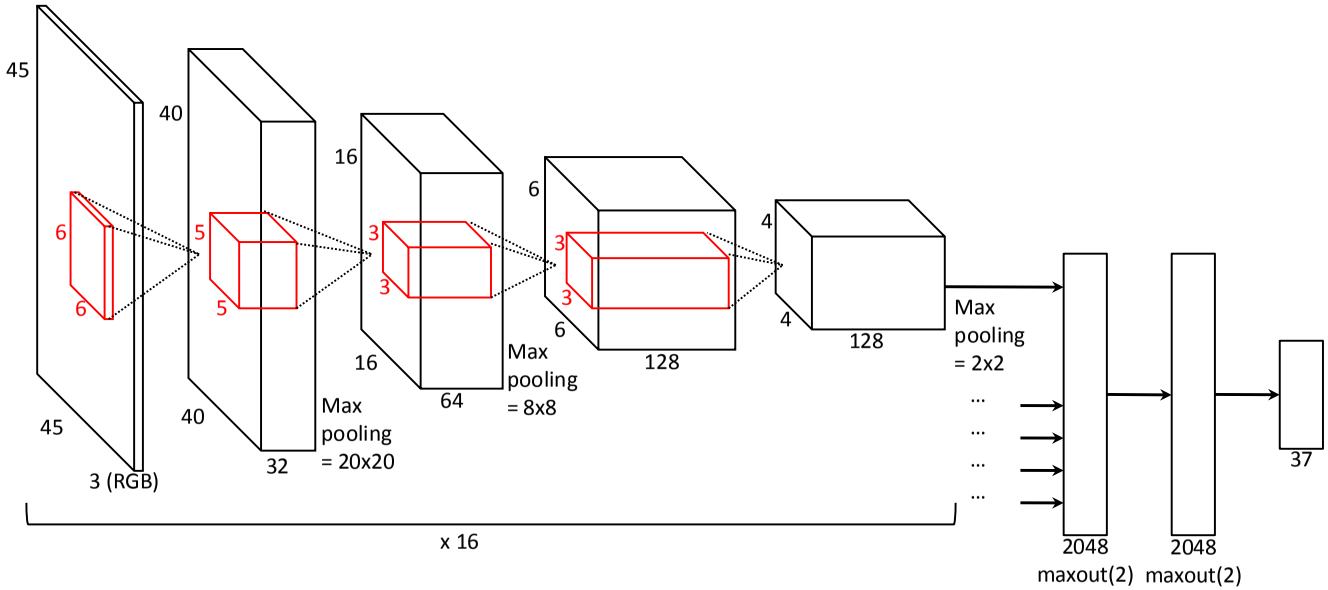


Figure A.6: Schematic overview of the architecture of the best performing network that we trained. The sizes of the filters and feature maps are indicated for each layer.

	type	# features	filter size	non-linearity	initial biases	initial weights
1	convolutional	32	6×6	ReLU	0.1	$\mathcal{N}(0, 0.01)$
2	convolutional	64	5×5	ReLU	0.1	$\mathcal{N}(0, 0.01)$
3	convolutional	128	3×3	ReLU	0.1	$\mathcal{N}(0, 0.01)$
4	convolutional	128	3×3	ReLU	0.1	$\mathcal{N}(0, 0.1)$
5	dense	2048	–	maxout (2)	0.01	$\mathcal{N}(0, 0.001)$
6	dense	2048	–	maxout (2)	0.01	$\mathcal{N}(0, 0.001)$
7	dense	37	–	constraints	0.1	$\mathcal{N}(0, 0.01)$

Table A.2: The hyperparameters of the trainable layers of the best performing network that we trained, also depicted in Figure A.6. The last two columns describe the initialization distributions of the weights and biases of each layer. See Section A.6.6 for a description of the incorporation of the output constraints into the last layer of the network.

it consists only of differentiable operations[9], the gradient of the objective function can be backpropagated through it. This guarantees that the output of the network will not violate the constraints that the answer probabilities must adhere to (for example, p_{bar} must be greater to or equal to p_{spiral} in the cumulative probabilities, since it is a higher-level question in the decision tree). This resulted in a small but significant performance improvement.

In addition to the best performing network, we also trained variants for the purpose of model averaging (see Section A.6.8). These networks differ slightly from the best performing network, and make slightly different predictions as a result. Variants included:

- a network with only two dense layers instead of three;
- a network with a different filter size configuration (filter sizes 8, 4, 3, 3 respectively instead of 6, 5, 3, 3);
- a network with ReLUs in the dense layers instead of maxout units;
- a network with 256 filters instead of 128 in the topmost convolutional layer.

In total, 17 different networks were trained on this data set.

A.6.7 Training

To train the models we used minibatch gradient descent with a batch size[10] of 16 and *Nesterov momentum* [11] with coefficient $\mu = 0.9$. Nesterov momentum is a method for accelerating gradient descent by accumulating gradients over time in directions that consistently decrease the objective function value. This and similar methods have are commonly used neural network training because they speed up the training process and often lead to improved predictive performance [144].

We performed approximately 1.5 million gradient updates, corresponding to 25 million training examples. Following Krizhevsky et al. [73], we used a discrete learning rate schedule to improve convergence. We began with a constant learning rate $\eta = 0.04$ and decreased it tenfold twice: it was decreased to 0.004 after 18 million examples, and to 0.0004 after 23 million examples. For the first 10,000 examples, the output constraints were ignored,

[9] Although the rectification operation is not technically differentiable everywhere, it is subdifferentiable so this does not pose a problem in practice.

[10] The batch size chosen is small because the convolutional part of the network is applied 16 times to different viewpoints of the input images, yielding an effective batch size of 256.

and the linear output of the top layer of the network was simply clipped between 0 and 1. This was necessary to ensure convergence.

Weights in the model were initialized by sampling from zero-mean normal distributions [9]. The variances of these distributions were fixed at each layer, and were manually chosen to ensure proper flow of the gradient through the network. All biases were initialized to positive values to decrease the risk of units getting stuck in the saturation region. Although this is not necessary for maxout units, the same strategy was used for the dense layers. The initialization strategy for all layers is shown in the last two columns of Table A.2.

During training, we used dropout [57] in all three dense layers. Using dropout was essential to reduce overfitting to manageable levels.

A.6.8 Model averaging

To further improve the prediction accuracy, we averaged the predictions of several different models, and across several transformations of the input images. Two requirements for model averaging to be effective is that each individual model must have roughly the same prediction accuracy, and the prediction errors should be as uncorrelated as possible.

For each model, we computed predictions for 60 affine transformations of the input images: a combination of 10 rotations, spaced by 36°, 3 rescalings (with scale factors 1.2^{-1}, 1 and 1.2) and optional horizontal flipping. An unweighted average of the predictions was computed. Even though the model is trained to be robust to these types of deformations (see Section A.6.4), computing averaged predictions in this fashion still helped to increase prediction accuracy (see Table A.3).

In total, 17 variants of the model were trained with predictions computed from the mean across 60 transformations. This resulted in 1020 sets of predictions averaged in total.

A.6.9 Implementation

All aspects of the model were implemented using Python and the Theano library [14, 7]. This allowed the use of GPU acceleration without any additional effort. Theano is also able to perform automatic differentiation, which simplifies the implementation of gradient-based optimization techniques. Networks were trained on NVIDIA GeForce GTX 680 cards. Data augmentation was performed on the CPU using the `scikit-image` package [151] in parallel with model training on the GPU. Training the network described in Section A.6.6 took roughly 67 hours in real time.

model	leaderboard score	
	public	private
best performing network	0.07671	0.07693
+ averaging over 60 transformations	0.07579	0.07603
+ averaging over 17 networks	0.07467	0.07492

Table A.3: Performance (in RMSE) of the best performing network, as well as the performance after averaging across 60 transformations of the input, and across 17 variants of the network. Please refer to Section A.3 for details on how the scores were computed.

The code to reproduce the winning submission for the Galaxy Challenge is available at `https://github.com/benanne/kaggle-galaxies`.

A.7 Results

Competition results of the models are listed in Table A.3. We report the performance of our best performing network, with and without averaging across 60 transformations, as well as that of the combination of all 17 variants. The root-mean-square error in Table A.3 is the same metric used to score submissions in the Galaxy Challenge (Equation A.1). Both averaging across transformations and averaging across different models contributed significantly to the final score. It is worth noting that our model performs well even without any model averaging, which is important because fast inference is desirable for practical applications. If predictions are to be generated for millions of images, combining a large number of predictions for each image would require an impractical amount of computation.

Although morphology prediction was framed as a regression problem in the competition (see Section A.3), it is fundamentally a classification task. To demonstrate the capabilities of our model in a more interpretable fashion, we can look at classification accuracies. For each question, we can obtain classifications by selecting the answer with the highest probability for each image. We can do this both for the probabilities obtained from Galaxy Zoo participants, and for the probabilities predicted by our model. We can then compute the classification accuracy simply by counting the number of images for which the classifications match up. Reducing the probability distributions to classifications in this fashion clearly causes some information to be discarded, but classification accuracy is a metric that is much easier to interpret.

To find out how the level of agreement between the Galaxy Zoo partici-

pants affects the accuracy of the predictions of our model, we can compute the entropy of the probability distribution over the answers for a given question. The entropy of a discrete probability distribution p over n options $x_1, \ldots, x_n$ is given by:

$$H(p) = -\sum_{i=1}^{n} p(x_i) \log p(x_i). \tag{A.2}$$

If the entropy is minimal, all participants selected the same answer (i.e. everyone agreed). If the entropy is maximal, all answers were equally likely to be selected. The entropy ranges between 0 and $\log(n)$. We can convert it into a measure of agreement $a(p)$ as follows:

$$a(p) = 1 - \frac{H(p)}{\log(n)}. \tag{A.3}$$

The quantity $a(p)$ will equal 0 in case of maximal disagreement, and 1 in case of maximal agreement.

To assess the conditions under which the predictions of the model can be trusted, we can measure the confidence of a prediction using the same measure $a(p)$ by applying it to the probability distributions predicted by the model, instead of the distributions of the crowdsourced answers. This allows us to relate model confidence and prediction accuracy.

For each question, we selected the subset of images from the real-time evaluation set[11] for which at least 50% of participants answered the question. This is to ensure that we only consider images for which the question is likely relevant. We ranked all images in this subset according to the measure $a(p)$ and divided them into 10 equal bins. We did this seperately for both the crowdsourced answers and the model predictions. For each bin, we computed the average of $a(p)$ and the classification accuracy using the best performing network (no averaging). These values are visualized in a set of graphs for each question in Figure A.7. The red circles show classification accuracy versus agreement. The blue squares show classification accuracy versus model confidence. The classification accuracy across the entire subset is also shown as a thick horizontal line. The dashed horizontal lines indicate the maximal accuracy of 100% and the chance-level accuracy, which depends on the number of options. The number of images in each subset and the overall classification accuracy are indicated above the graphs.

For all questions, the classification accuracy tapers off as the level of agreement between Galaxy Zoo participants decreases. This makes sense,

[11] We could also have conducted this analysis on the evaluation set from the competition, but since the true answer probabilities for the real-time evaluation set were readily available and this set contains over 6,000 images, we used this instead.

as those images are harder to classify. Kuminski et al. [76] report similar results using the WND-CHARM algorithm, with lowest accuracies for features describing spiral arm and irregular structures. Our model achieves near-perfect accuracy for most of the questions when the level of agreement is high. Classifications for bulge dominance (Q5) and spiral arm tightness (Q10) have low agreement overall, and are also more difficult to answer for the model.

Similarly, the confidence of the model in its predictions is correlated with classification accuracy: we achieve near-perfect accuracy for most questions when the model is highly confident. This is a useful property, because it allows us to determine when we are able to trust the predictions, and when we should defer to an expert instead. As a consequence, the model could be used to filter a large collection of images, in order to obtain a much smaller set that can be annotated manually by experts. Such a two-stage approach would greatly reduce the experts' workload at virtually no cost in terms of accuracy.

For questions 1, 2, 3, 6 and 7 in particular, we are able to make confident, accurate predictions for the majority of examples. This would allow us to largely automate the assessment of e.g. smoothness (Q1) and roundedness (Q7). For questions 5 and 10 on the other hand, confidence is low across the board and the classification accuracy is usually too low to be of practical use. As a result, determining bulge dominance (Q5) and spiral arm tightness (Q10) would still require a lot of manual input. The level to which we are able to automate the annotation process depends on the morphological properties we are interested in, as well as the distribution of morphology types in the dataset we wish to analyse.

To assess how well the model is able to predict various different morphology types, we computed precision and recall scores for all answers individually. The precision (P) and recall (R) scores are defined in terms of the number of true positive (TP), false positive (FP) and false negative (FN) classifications as follows:

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}. \tag{A.4}$$

The scores are listed in Table A.4. We used the same strategy as before to obtain classifications, and only considered those examples for which at least 50% of the Galaxy Zoo participants answered the question. The numbers of examples that were available for each question and answer are also shown.

From these scores, we can establish that the model has more difficulty with morphology types that occur less frequently in the dataset, e.g., *star or artifact* (A1.3), *no bulge* (A5.1), *dominant bulge* (A5.4) and *dust lane* (A8.7). We note that images in the first category are attempted to be deli-

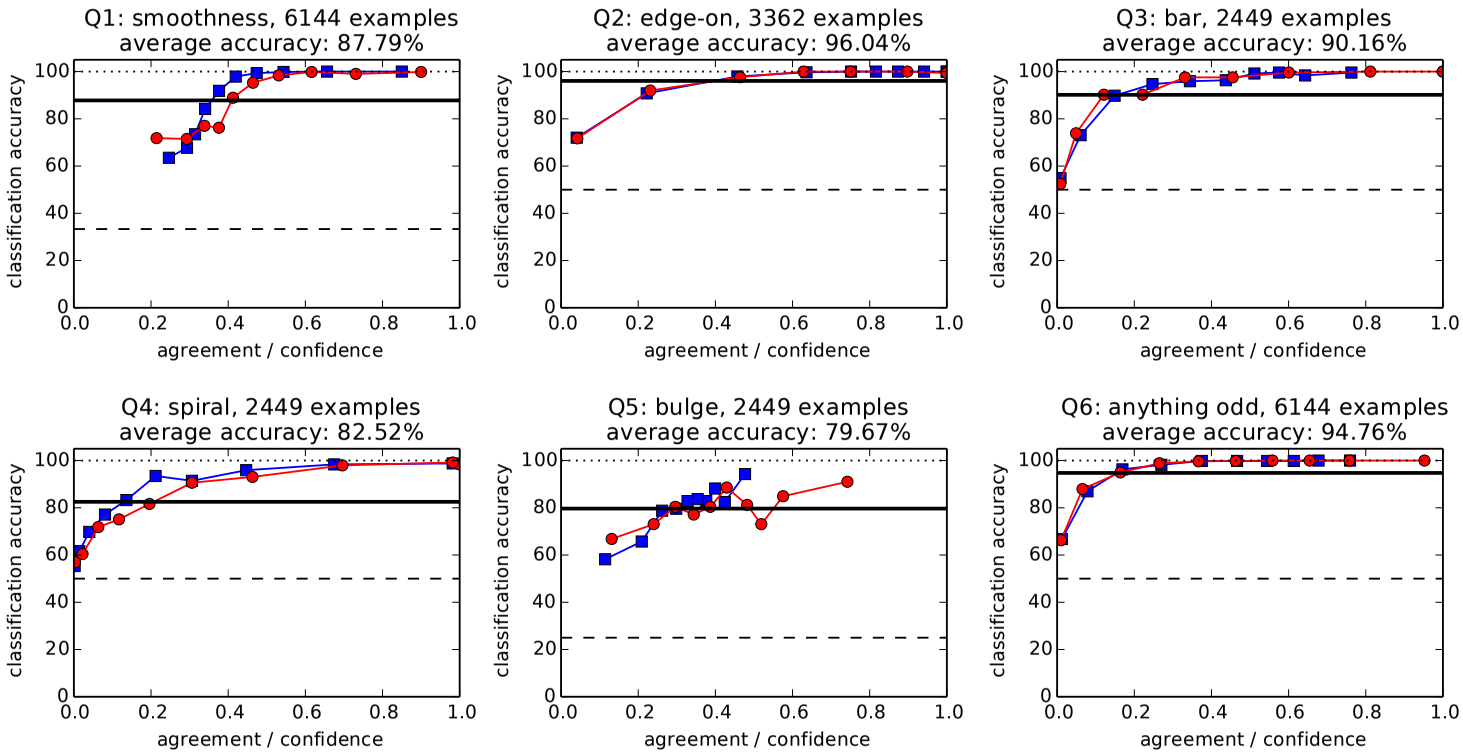
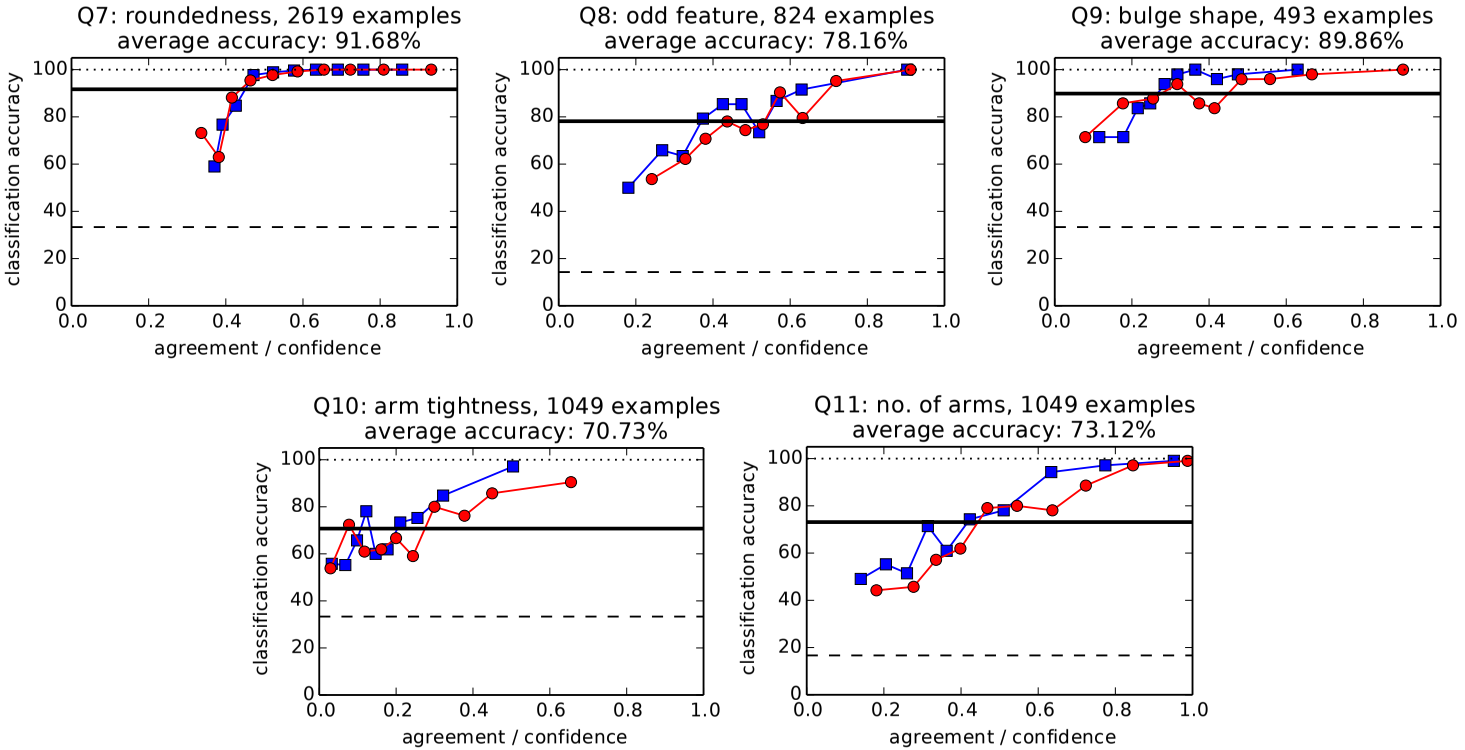


Figure A.7: Level of agreement (red circles) and model confidence (blue squares) versus classification accuracy for all questions (see Table A.1), computed on the real-time evaluation set. The overall classification accuracy is indicated as a thick horizontal line. The dotted and dashed horizontal lines indicate the maximal accuracy of 100% and the chance-level accuracy respectively. The number of images that were included in the analysis and the overall classification accuracy for each question are indicated above the graphs.


Q7: roundedness, 2619 examples
average accuracy: 91.68%
Q8: odd feature, 824 examples
average accuracy: 78.16%
Q9: bulge shape, 493 examples
average accuracy: 89.86%
Q10: arm tightness, 1049 examples
average accuracy: 70.73%
Q11: no. of arms, 1049 examples
average accuracy: 73.12%
classification accuracy
agreement / confidence
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0

brately excluded from the Galaxy Zoo data set via flags in the SDSS pipeline. Both the precision and recall scores are affected, so this effect cannot be attributed entirely to a bias towards more common morphologies. However, recall is generally affected more strongly than precision, which indicates that the model is more conservative in predicting rare morphology types. For a few very rare answers, we were unable to compute precision scores because the model never predicted them for the examples that were considered: *lens or arc* (A8.2), *boxy bulge* (A9.2) and *four spiral arms* (A11.4). While these are all rare morphologies, they have considerable scientific interest and constructing a model that can accurately identify them is still a primary goal.

A.8 Analysis

Traditionally, neural networks are often treated as black boxes that perform some complicated and uninterpretable sequence of computations that yield a good approximation to the desired output. However, analysing the parameters of a trained model can be very informative, and sometimes even leads to new insights about the problem the network is trying to solve [160]. This is especially true for convolutional neural networks trained on images, where the first-layer filters can be interpreted visually.

Figure A.8 shows the 32 filters learned in the first layer of the best performing network described in Section A.6.6. Each filter was contrast-normalized individually to bring out the details, and the three colour channels are shown separately. Comparing the filter weights across colour channels reveals that some filters are more sensitive to particular colours, while others are sensitive to patterns, edges and textures. The same phenomenon is observed when training convolutional neural networks on more traditional image datasets. The filters for edge detection seem to be looking for curved edges in particular, which is to be expected because of the radial symmetry of the input images.

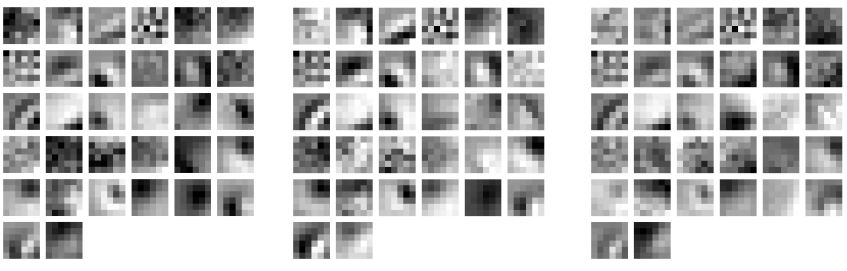
It is also possible to visualize what neurons in the topmost hidden layer of the network (i.e. just before the output layer) have learned about the data, by selecting representative examples from the test set that maximize their activations. This reveals what type of inputs the unit is sensitive to, and what kind of invariances it has learned. Because we used maxout units in this layer, we can also select examples that minimally activate the units, allowing us to determine which types of inputs each unit discriminates between.

Figure A.9 shows such a visualization for three different units. Clearly each unit is able to discriminate between two distinct types of galaxies. The units also exhibit rotation invariance, as well as some scale invariance. For

		precision	recall	# examples
Q1: smoothness				6144
A1.1	smooth	0.8459	0.8841	2700
A1.2	features or disk	0.9051	0.8742	3435
A1.3	star or artifact	1.0000	0.4444	9
Q2: edge-on				3362
A2.1	yes	0.9065	0.8885	655
A2.2	no	0.9732	0.9778	2707
Q3: bar				2449
A3.1	yes	0.7725	0.7101	483
A3.2	no	0.9302	0.9486	1966
Q4: spiral				2449
A4.1	yes	0.8715	0.8270	1451
A4.2	no	0.7659	0.8226	998
Q5: bulge				2449
A5.1	no bulge	0.6697	0.5000	146
A5.2	just noticeable	0.7828	0.8475	1174
A5.3	obvious	0.8292	0.8049	1092
A5.4	dominant	0.4444	0.1081	37
Q6: anything odd				6144
A6.1	yes	0.8438	0.7500	828
A6.2	no	0.9617	0.9784	5316
Q7: roundedness				2619
A7.1	completely round	0.9228	0.9282	1197
A7.2	in between	0.9128	0.9171	1279
A7.3	cigar-shaped	0.9000	0.8182	143

Table A.4: Precision and recall scores for each answer. We compute these values only for the subset of examples in the real-time evaluation set where at least 50% of participants answered the question. We also give the number of examples that are in this subset for each answer. A question mark indicates that we were unable to compute the precision score because the model did not predict this answer for any of the considered examples.

		precision	recall	# examples
Q8: odd feature				824
A8.1	ring	0.9097	0.9161	143
A8.2	lens or arc	?	0.0000	2
A8.3	disturbed	0.8000	0.4138	29
A8.4	irregular	0.8579	0.8674	181
A8.5	other	0.6842	0.6810	210
A8.6	merger	0.7398	0.7773	256
A8.7	dust lane	0.5000	0.6667	3
Q9: bulge shape				493
A9.1	rounded	0.9143	0.9412	340
A9.2	boxy	?	0.0000	8
A9.3	no bulge	0.8601	0.8483	145
Q10: arm tightness				1049
A10.1	tight	0.7500	0.7350	449
A10.2	medium	0.6619	0.7112	457
A10.3	loose	0.7373	0.6084	143
Q11: no. of arms				1049
A11.1	1	1.0000	0.2037	54
A11.2	2	0.8201	0.8691	619
A11.3	3	0.4912	0.3182	88
A11.4	4	?	0.0000	21
A11.5	more than 4	0.4000	0.4000	20
A11.6	can't tell	0.5967	0.7368	247



(a) red channel **(b)** green channel **(c)** blue channel

Figure A.8: The 32 filters learned in the first convolutional layer of the best-performing network. Each filter was contrast-normalized individually across all channels.

some units, we observed selectivity only in the positive or in the negative direction (not shown). A minority of units seem to be multimodal, activating in the same direction for two or more distinct types of galaxies. Presumably the activation value of these units is disambiguated in the context of all other unit values.

The unit visualized in Figure A.9b detects imaging artifacts: black lines running across the center of the images, which are the result of dead pixels in the SDSS camera. This is interesting because such (known) artifacts are not morphological features of the depicted galaxies. It turns out that the network is trying to replicate the behaviour of the Galaxy Zoo participants, who tend to classify images featuring such artifacts as *disturbed* galaxies (answer A8.3 in Table A.1), even though this is not the intended meaning of this answer. Most likely this is because the button for this answer in the Galaxy Zoo 2 web interface seems to feature such a black line.

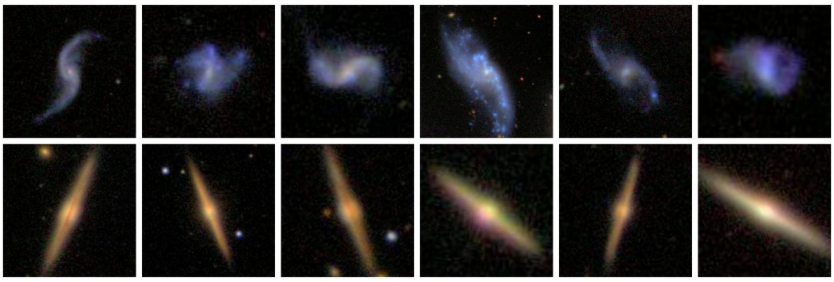
Finally, we can look at some examples from the real-time evaluation set (see Section A.6.1) with low and high prediction errors, to get an idea of the strengths and weaknesses of the model (Figure A.10). The reported RMSE values were obtained with the best performing network and without any averaging, and without centering or rescaling.

The images that are difficult to classify are quite varied. Some are faint, but look fairly typical otherwise, such as Figure A.10a. Most are negatively affected by the cropping operation in various ways: either because they are not properly centered, or because they are very large (Figures A.10b and A.10c respectively). This was the original motivation for introducing an additional rescaling and centering step during preprocessing, but did not end up improving the overall prediction accuracy. The easiest galaxies to classify are mostly smooth, round ellipticals.

A.9 Conclusion and future work

We presented a convolutional neural network for fine-grained galaxy morphology prediction, with a novel architecture that allows us to exploit rotational symmetry in the input images. The network was trained on data from the Galaxy Zoo 2 project and is able to reliably predict various aspects of galaxy morphology directly from raw pixel data, without requiring any form of handcrafted feature extraction. It can automatically annotate large collections of images, enabling quantitative studies of galaxy morphology on an unprecedented scale.

Our novel approach to exploiting rotational symmetry was essential to achieve state-of-the-art performance, winning the Galaxy Challenge hosted

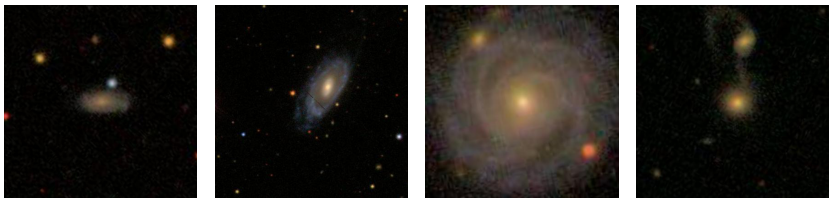


(a)

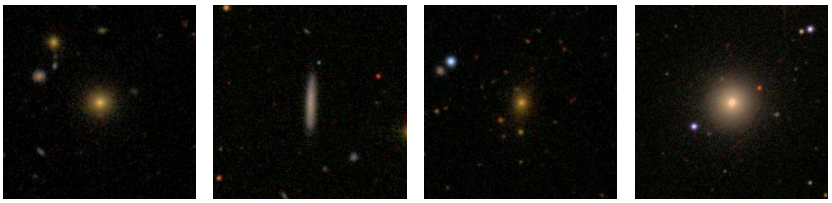
(b)

(c)

Figure A.9: Example images from the test set that maximally and minimally activate units in the topmost hidden layer of the best performing network. Each group of 12 images represents a unit. The top row of images in each group maximally activate the unit, and bottom row of images minimally activate it. From top to bottom, these galaxies primarily correspond to the Galaxy Zoo 2 labels of: loose winding arms, edge-on disks, irregulars, disturbed, other, and tight winding arms.



(a) 0.24610 **(b)** 0.21877 **(c)** 0.21145 **(d)** 0.20088



(e) 0.01112 **(f)** 0.01174 **(g)** 0.01187 **(h)** 0.01223

Figure A.10: Example images from the real-time evaluation set, along with their prediction RMSEs for the best-performing network (lower is better). The images on the top row were the most difficult for the model to classify; the images on the bottom row were the easiest. Larger angular size and non-radially symmetric morphology are the most challenging targets for the model.

on Kaggle. Although our winning solution required averaging many sets of predictions from different networks for each image, using a single network also yields competitive results.

Our model can be adapted to work with any collection of centered galaxy images and arbitrary morphological decision trees. Our implementation was developed using open source tools and the source code is publicly available. The model can be trained and used on consumer hardware. Its predictions are highly reliable when they are confident, making our approach applicable for fine-grained morphological analysis of large-scale survey data. Performing such large-scale analyses is an important direction for future research.

In the future, it would be interesting to train networks on larger collections of annotated images. From previous applications in the domain of computer vision, it has become clear that the performance of convolutional neural networks scales very well with the size of the dataset. The set of $\sim 55,000$ galaxy images used in this paper (90% of the provided training set) is quite a small dataset by modern standards. Even though we combined several techniques to avoid overfitting, which allowed us to train very large models on this dataset effectively, a clear opportunity to improve predictive performance is to train the same model on a larger dataset, since Galaxy Zoo has already collected annotations for a much larger number of images. More recent iterations of the Galaxy Zoo project have concentrated on higher redshift samples, so care will have to be taken to ensure that the model is able to generalize across different redshift slices.

The use of larger datasets may also allow for a further increase in model capacity (i.e. the number of trainable parameters) without the risk of excessive overfitting. These high-capacity models could be used as the basis for much larger surveys such as the LSST (Large Synoptic Survey Telescope). The integration of model predictions into existing annotation workflows, both by experts and through crowdsourcing platforms, will also require further study.

Another possibility is the application of our approach to raw photometric data which have not been preprocessed for visual inspection by humans. The networks should be able to learn useful features from this representation, including structural changes from multiple wavebands [eg, 55]. Automated classification of other data modalities that exhibit radial symmetry (a commonly occurring property in nature, e.g. in flowers, animals) also presents an interesting opportunity.

From a machine learning point of view, it would be useful to investigate improved network architectures based on recent developments, such as the trend towards deeper networks with in excess of 20 layers of processing and the use of smaller receptive fields [145, 132].

Bibliography

[1] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, and Gerald Penn. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4277–4280. IEEE, 2012.

[2] Mike Tyka Alexander Mordvintsev, Christopher Olah. Inceptionism: Going deeper into neural networks, 2015.

[3] Joakim Andén and Stéphane Mallat. Multiscale scattering for audio classification. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, 2011.

[4] Nicholas M Ball, Jon Loveday, Masataka Fukugita, Osamu Nakamura, Sadanori Okamura, Jon Brinkmann, and Robert J Brunner. Galaxy types in the sloan digital sky survey using supervised artificial neural networks. *Monthly Notices of the Royal Astronomical Society*, 348(3): 1038–1046, 2004.

[5] Steven P Bamford, Robert C Nichol, Ivan K Baldry, Kate Land, Chris J Lintott, Kevin Schawinski, Anze Slosar, Alexander S Szalay, Daniel Thomas, Mehri Torki, et al. Galaxy zoo: the dependence of morphology and colour on environment. *Monthly Notices of the Royal Astronomical Society*, 393(4):1324–1352, 2009.

[6] Manda Banerji, Ofer Lahav, Chris J Lintott, Filipe B Abdalla, Kevin Schawinski, Steven P Bamford, Dan Andreescu, Phil Murray, M Jordan Raddick, Anze Slosar, et al. Galaxy zoo: reproducing galaxy morphologies via machine learning. *Monthly Notices of the Royal Astronomical Society*, 406(1):342–353, 2010.

[7] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*, 2012.

[8] Yoshua Bengio. Learning deep architectures for AI. Technical report, Dept. IRO, Université de Montreal, 2007.

[9] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478. Springer, 2012.

[10] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 2012.

[11] Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. Advances in optimizing recurrent networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8624–8628. IEEE, 2013.

[12] James Bennett and Stan Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[13] J Bergstra, D Yamins, and DD Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 115–123, 2013.

[14] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010.

[15] E Bertin. Classification of astronomical images with a neural network. In *Science with Astronomical Near-Infrared Sky Surveys*, pages 49–51. Springer, 1994.

[16] Emmanuel Bertin and S Arnouts. Sextractor: Software for source extraction. *Astronomy and Astrophysics Supplement Series*, 117:393–404, 1996.

[17] Thierry Bertin-Mahieux, Ron J. Weiss, and Daniel P.W. Ellis. Clustering beat-chroma patterns in a large music database. In *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR)*, 2010.

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

[19] Christopher M Bishop. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[20] Y-Lan Boureau, Jean Ponce, and Yann Lecun. A theoretical analysis of feature pooling in visual recognition. In *27th International Conference on Machine Learning*, 2010.

[21] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[22] P.J. Burt and E.H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31:532–540, 1983. ISSN 0090-6778.

[23] Michael A. Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. Content-based music information retrieval: Current directions and future challenges. In *Proceedings of the IEEE*, volume 96, pages 668–696, April 2008.

[24] Ò. Celma. *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, 2008.

[25] Daniel Clery. Galaxy Zoo Volunteers Share Pain and Glory of Research. *Science*, 333(6039):173–175, July 2011. doi: 10.1126/science.333.6039.173. URL `http://dx.doi.org/10.1126/science.333.6039.173`.

[26] Adam Coates and Andrew Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 2011.

[27] Adam Coates and Andrew Y. Ng. Learning feature representations with k-means. *Neural Networks: Tricks of the Trade, Reloaded*, 2012.

[28] Adam Coates, Andrew Y. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. *Journal of Machine Learning Research - Proceedings Track*, 15:215–223, 2011.

[29] Adrian A Collister and Ofer Lahav. Annz: estimating photometric redshifts using artificial neural networks. *Publications of the Astronomical Society of the Pacific*, 116(818):345–351, 2004.

[30] DW Darg, S Kaviraj, CJ Lintott, K Schawinski, M Sarzi, Steven Bamford, J Silk, R Proctor, D Andreescu, P Murray, et al. Galaxy zoo: the fraction of merging galaxies in the sdss and their morphologies. *Monthly Notices of the Royal Astronomical Society*, 401(2):1043–1056, 2010.

[31] Jorge De La Calleja and Olac Fuentes. Machine learning and image analysis for morphological galaxy classification. *Monthly Notices of the Royal Astronomical Society*, 349(1):87–93, 2004.

[32] Sander Dieleman and Benjamin Schrauwen. Multiscale approaches to music audio feature learning. In *Proceedings of the 14th International Conference on Music Information Retrieval (ISMIR)*, 2013.

[33] Sander Dieleman, Philémon Brakel, and Benjamin Schrauwen. Audio-based music classification with a pretrained convolutional network. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, 2011.

[34] Sander Dieleman, Kyle W Willett, and Joni Dambre. Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly Notices of the Royal Astronomical Society*, 450(2): 1441–1459, 2015.

[35] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. pages 153–160, April 2009.

[36] F. Eyben, S. Böck, B. Schuller, and A. Graves. Universal onset detection with bidirectional long-short term memory neural networks. In *Proc. 11th Intern. Soc. for Music Information Retrieval Conference, ISMIR, Utrecht, The Netherlands, 2010*, 2010.

[37] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, August 2013. in press.

[38] Andrew E Firth, Ofer Lahav, and Rachel S Somerville. Estimating photometric redshifts with artificial neural networks. *Monthly notices of the royal astronomical society*, 339(4):1195–1202, 2003.

[39] SR Folkes, O Lahav, and SJ Maddox. An artificial neural network approach to the classification of galaxy spectra. *Monthly Notices of the Royal Astronomical Society*, 283(2):651–665, 1996.

[40] Jonathan T Foote. Content-based retrieval of music and audio. In *Voice, Video, and Data Communications*, pages 138–147. International Society for Optics and Photonics, 1997.

[41] Rémi Foucard, Slim Essid, Mathieu Lagrange, and Gaël Richard. Multi-scale temporal fusion by boosting for music classification. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, 2011.

[42] Zhouyu Fu, Guojun Lu, Kai Ming Ting, and Dengsheng Zhang. A survey of audio-based music classification and annotation. *IEEE Transactions on Multimedia*, 13(2):303–319, 2011. ISSN 1520-9210.

[43] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[44] Robert Gens and Pedro Domingos. Deep symmetry networks. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, 2014.

[45] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *JMLR W&CP: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011)*, 2011.

[46] Hanlin Goh, Nicolas Thome, and Matthieu Cord. Biasing restricted boltzmann machines to manipulate latent selectivity and sparsity. In *Deep Learning and Unsupervised Feature Learning Workshop — NIPS*, 2010.

[47] Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems 17*, pages 513–520, 2004.

[48] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[49] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

[50] Simone Gori, Kai Hamburger, and Lothar Spillmann. Reversal of apparent rotation in the enigma-figure with and without motion adaptation and the effect of t-junctions. *Vision research*, 46(19):3267–3273, 2006.

[51] Philippe Hamel and Douglas Eck. Learning features from music audio with deep belief networks. In *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR)*, 2010.

[52] Philippe Hamel, Sean Wood, and Douglas Eck. Automatic identification of instrument classes in polyphonic and poly-instrument audio. In Keiji Hirata, George Tzanetakis, and Kazuyoshi Yoshii, editors, *ISMIR*, pages 399–404. International Society for Music Information Retrieval, 2009. ISBN 978-0-9813537-0-8. URL `http://dblp.uni-trier.de/db/conf/ismir/ismir2009.html#HamelWE09`.

[53] Philippe Hamel, Simon Lemieux, Yoshua Bengio, and Douglas Eck. Temporal pooling and multiscale learning for automatic annotation and ranking of music audio. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, 2011.

[54] Philippe Hamel, Yoshua Bengio, and Douglas Eck. Building musically-relevant audio features through multiple timescale representations. In *Proceedings of the 13th International Conference on Music Information Retrieval (ISMIR)*, 2012.

[55] B. Häußler, S. P. Bamford, M. Vika, A. L. Rojas, M. Barden, L. S. Kelvin, M. Alpaslan, A. S. G. Robotham, S. P. Driver, I. K. Baldry, S. Brough, A. M. Hopkins, J. Liske, R. C. Nichol, C. C. Popescu, and R. J. Tuffs. MegaMorph - multiwavelength measurement of galaxy structure: complete Sérsic profile information from modern surveys. *MNRAS*, 430:330–369, March 2013. doi: 10.1093/mnras/sts633.

[56] Mikael Henaff, Kevin Jarrett, Koray Kavukcuoglu, and Yann LeCun. Unsupervised learning of sparse features for scalable audio classification. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, 2011. ISBN 978-0-615-54865-4.

[57] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. Technical report, University of Toronto, 2012.

[58] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke,

Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

[59] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:2002, 2000.

[60] Geoffrey E. Hinton. A practical guide to training restricted boltzmann machines. Technical report, University of Toronto, 2010.

[61] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006. ISSN 0899-7667. doi: http://dx.doi.org/10.1162/neco.2006.18.7.1527.

[62] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[63] Matthew Hoffman, David Blei, and Perry Cook. Easy As CBA: A Simple Probabilistic Model for Tagging Music. In *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR)*, 2009. URL `http://ismir2009.ismir.net/proceedings/OS5-2.pdf`.

[64] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, 2008. ISBN 978-0-7695-3502-9.

[65] Marc Huertas-Company, JA Aguerri, M Bernardi, S Mei, and J Sánchez Almeida. Revisiting the hubble sequence in the sdss dr7 spectroscopic sample: a publicly available bayesian automated classification. *arXiv preprint arXiv:1010.3018*, 2010.

[66] Eric J. Humphrey, Aron P. Glennon, and Juan Pablo Bello. Non-linear semantic embedding for organizing large instrument sample libraries. In Xue wen Chen, Tharam S. Dillon, Hisao Ishbuchi, Jian Pei, Haixun Wang, and M. Arif Wani, editors, *ICMLA (2)*, pages 142–147. IEEE Computer Society, 2011. URL `http://dblp.uni-trier.de/db/conf/icmla/icmla2011-2.html#HumphreyGB11`.

[67] Eric J. Humphrey, Juan P. Bello, and Yann LeCun. Moving beyond feature design: Deep architectures and automatic feature learning in music informatics. In *Proceedings of the 13th International Conference on Music Information Retrieval (ISMIR)*, 2012.

[68] Aapo Hyvärinen and Patrik Hoyer. Emergence of phase- and shift-invariant features by decomposition of natural images into independent feature subspaces. *Neural Comput.*, 12(7):1705–1720, July 2000. ISSN 0899-7667. doi: 10.1162/089976600300015312. URL `http://dx.doi.org/10.1162/089976600300015312`.

[69] Navdeep Jaitly and Geoffrey E. Hinton. Learning a better representation of speech soundwaves using restricted boltzmann machines. In *Proceedings of ICASSP 2011*, pages 5884–5887, 2011.

[70] Tristan Jehan and David DesRoches. The echo nest analyzer documentation, January 2014. URL `http://developer.echonest.com/docs/v4/_static/AnalyzeDocumentation.pdf`.

[71] Koray Kavukcuoglu, Pierre Sermanet, Y-Lan Boureau, Karol Gregor, Michaël Mathieu, and Yann L Cun. Learning convolutional feature hierarchies for visual recognition. In *Advances in neural information processing systems*, pages 1090–1098, 2010.

[72] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[73] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, 2012.

[74] Alex Krizhevsky. Convolutional deep belief networks on cifar-10. Technical report, University of Toronto, 2010.

[75] Richard Kronland-Martinet, Jean Morlet, and Alexander Grossmann. Analysis of sound patterns through wavelet transforms. *International Journal of Pattern Recognition and Artificial Intelligence*, 1(02):273–302, 1987.

[76] Evan Kuminski, Joe George, John Wallin, and Lior Shamir. Combining human and machine learning for morphological analysis of galaxy images. *Publications of the Astronomical Society of the Pacific*, 126 (944):959–967, 2014.

[77] Alexandre Lacoste and Douglas Eck. A supervised classification algorithm for note onset detection. In *EURASIP Journal on Applied Signal Processing*, 2007.

[78] O. Lahav, A. Naim, R. J. Buta, H. G. Corwin, G. de Vaucouleurs, A. Dressler, J. P. Huchra, S. van den Bergh, S. Raychaudhury, L. Sodre, Jr., and M. C. Storrie-Lombardi. Galaxies, Human Eyes, and

Artificial Neural Networks. *Science*, 267:859–862, February 1995. doi: 10.1126/science.267.5199.859.

[79] O Lahav, A Nairn, L Sodré, and MC Storrie-Lombardi. Neural computation as a tool for galaxy classification: methods and examples. *Monthly Notices of the Royal Astronomical Society*, 283(1):207–221, 1996.

[80] K. Land, A. Slosar, C. Lintott, D. Andreescu, S. Bamford, P. Murray, R. Nichol, M. J. Raddick, K. Schawinski, A. Szalay, D. Thomas, and J. Vandenberg. Galaxy Zoo: the large-scale spin statistics of spiral galaxies in the Sloan Digital Sky Survey. *MNRAS*, 388:1686–1692, August 2008. doi: 10.1111/j.1365-2966.2008.13490.x.

[81] Edith Law and Luis von Ahn. Input-agreement: a new mechanism for collecting data using human computation games. In *Proceedings of the 27th international conference on Human factors in computing systems*, 2009.

[82] Quoc V. Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeff Dean, and Andrew Y. Ng. Building high-level features using large scale unsupervised learning. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, 2012.

[83] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[84] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 609–616, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1. doi: http://doi.acm.org/10.1145/1553374.1553453. URL `http://doi.acm.org/10.1145/1553374.1553453`.

[85] Honglak Lee, Peter Pham, Yan Largman, and Andrew Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems 22*. 2009.

[86] Jong-Hwan Lee, Ho-Young Jung, Te-Won Lee, and Soo-Young Lee. Speech feature extraction using independent component analysis. In

Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on, volume 3, pages 1631–1634. IEEE, 2000.

[87] Michael S Lewicki. Efficient coding of natural sounds. *Nature neuroscience*, 5(4):356–363, 2002.

[88] TL Li, Antoni B Chan, and AH Chun. Automatic musical pattern feature extraction using convolutional neural network. In *Proc. Int. Conf. Data Mining and Applications*, 2010.

[89] Dawen Liang, Minshu Zhan, and Daniel P. W. Ellis. Content-aware collaborative music recommendation using pre-trained neural networks. In *Machine Learning for Music Discovery Workshop at the 32nd International Conference on Machine Learning*, 2015.

[90] C. Lintott, K. Schawinski, S. Bamford, A. Slosar, K. Land, D. Thomas, E. Edmondson, K. Masters, R. C. Nichol, M. J. Raddick, A. Szalay, D. Andreescu, P. Murray, and J. Vandenberg. Galaxy Zoo 1: data release of morphological classifications for nearly 900 000 galaxies. *MNRAS*, 410:166–178, January 2011. doi: 10.1111/j.1365-2966.2010.17432.x.

[91] C. J. Lintott, K. Schawinski, A. Slosar, K. Land, S. Bamford, D. Thomas, M. J. Raddick, R. C. Nichol, A. Szalay, D. Andreescu, P. Murray, and J. Vandenberg. Galaxy Zoo: morphologies derived from visual inspection of galaxies from the Sloan Digital Sky Survey. *MNRAS*, 389:1179–1189, September 2008. doi: 10.1111/j.1365-2966.2008.13689.x.

[92] Chris J Lintott, Kevin Schawinski, William Keel, Hanny Van Arkel, Nicola Bennert, Edward Edmondson, Daniel Thomas, Daniel JB Smith, Peter D Herbert, Matt J Jarvis, et al. Galaxy zoo:'hanny's voorwerp', a quasar light echo? *Monthly Notices of the Royal Astronomical Society*, 399(1):129–140, 2009.

[93] Beth Logan. Mel frequency cepstral coefficients for music modeling. In *International Symposium on Music Information Retrieval*, volume 28, page 5, 2000.

[94] R.F. Lyon. A computational model of filtering, detection, and compression in the cochlea. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '82.*, volume 7, pages 1282–1285, May 1982. doi: 10.1109/ICASSP.1982.1171644.

[95] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. *CoRR*, abs/1412.0035, 2014. URL `http://arxiv.org/abs/1412.0035`.

[96] Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks. *arXiv preprint arXiv:1406.3332*, 2014.

[97] Karen L Masters, Moein Mosleh, A Kathy Romer, Robert C Nichol, Steven P Bamford, Kevin Schawinski, Chris J Lintott, Dan Andreescu, Heather C Campbell, Ben Crowcroft, et al. Galaxy zoo: passive red spirals. *Monthly Notices of the Royal Astronomical Society*, 405(2): 783–799, 2010.

[98] Karen L Masters, Robert C Nichol, Ben Hoyle, Chris Lintott, Steven P Bamford, Edward M Edmondson, Lucy Fortson, William C Keel, Kevin Schawinski, Arfon M Smith, et al. Galaxy zoo: bars in disc galaxies. *Monthly Notices of the Royal Astronomical Society*, 411(3): 2026–2034, 2011.

[99] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[100] Brian McFee and Gert R. G. Lanckriet. Metric learning to rank. In *Proceedings of the 27 th International Conference on Machine Learning*, 2010.

[101] Brian McFee, Luke Barrington, and Gert R. G. Lanckriet. Learning content similarity for music recommendation. *IEEE Transactions on Audio, Speech & Language Processing*, 20(8), 2012.

[102] Brian McFee, Thierry Bertin-Mahieux, Daniel P.W. Ellis, and Gert R.G. Lanckriet. The million song dataset challenge. In *Proceedings of the 21st international conference companion on World Wide Web*, 2012. ISBN 978-1-4503-1230-1. doi: 10.1145/2187980.2188222. URL `http://doi.acm.org/10.1145/2187980.2188222`.

[103] T. Melvin, K. Masters, C. Lintott, R. C. Nichol, B. Simmons, S. P. Bamford, K. R. V. Casteels, E. Cheung, E. M. Edmondson, L. Fortson, K. Schawinski, R. A. Skibba, A. M. Smith, and K. W. Willett. Galaxy Zoo: an independent look at the evolution of the bar fraction over the last eight billion years from HST-COSMOS. *MNRAS*, January 2014. doi: 10.1093/mnras/stt2397.

[104] P. W. Mirowski, Y. LeCun, D. Madhavan, and R. Kuzniecky. Comparing svm and convolutional networks for epileptic seizure prediction

from intracranial eeg. *Machine Learning for Signal Processing, 2008. MLSP 2008. IEEE Workshop on*, pages 244–249, 2008.

[105] N. Morgan, Qifeng Zhu, A. Stolcke, K. Sonmez, S. Sivadas, T. Shinozaki, M. Ostendorf, P. Jain, H. Hermansky, D. Ellis, G. Doddington, B. Chen, O. Cretin, H. Bourlard, and M. Athineos. Pushing the envelope - aside [speech recognition]. *Signal Processing Magazine, IEEE*, 22(5):81–88, Sept 2005. ISSN 1053-5888. doi: 10.1109/MSP.2005.1511826.

[106] M. Muller, D.P.W. Ellis, A. Klapuri, and G. Richard. Signal processing for music analysis. *Selected Topics in Signal Processing, IEEE Journal of*, 5(6):1088–1110, 2011. ISSN 1932-4553.

[107] Meinard Müller. *Information Retrieval for Music and Motion*. Springer Verlag, 2007. ISBN 3540740473.

[108] Urs Muller, Jan Ben, Eric Cosatto, Beat Flepp, and Yann L Cun. Off-road obstacle avoidance through end-to-end learning. In *Advances in neural information processing systems*, pages 739–746, 2005.

[109] A Naim, O Lahav, L Sodre, and MC Storrie-Lombardi. Automated morphological classification of apm galaxies by supervised artificial neural networks. *Monthly Notices of the Royal Astronomical Society*, 275(3):567–590, 1995.

[110] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010.

[111] Juhan Nam, Jorge Herrera, Malcolm Slaney, and Julius O. Smith. Learning sparse feature representations for music annotation and retrieval. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, 2012.

[112] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. Multimodal deep learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[113] M. Norouzi, M. Ranjbar, and G. Mori. Stacks of convolutional restricted boltzmann machines for shift-invariant feature learning. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2735 –2742, 2009. doi: 10.1109/CVPR.2009.5206577.

[114] SC Odewahn, EB Stockwell, RL Pennington, RM Humphreys, and WA Zumach. Automated star/galaxy discrimination with neural networks. In *Digitised Optical Sky Surveys*, pages 215–224. Springer, 1992.

[115] Nikita Orlov, Lior Shamir, Tomasz Macura, Josiah Johnston, D Mark Eckley, and Ilya G Goldberg. Wnd-charm: Multi-purpose image classification using compound image transforms. *Pattern recognition letters*, 29(11):1684–1693, 2008.

[116] Kai Lars Polsterer, Fabian Gieseke, and Oliver Kramer. Galaxy classification without feature extraction. In *Astronomical Data Analysis Software and Systems XXI*, volume 461, page 561, 2012.

[117] Andreas Rauber, Alexander Schindler, and Rudolf Mayer. Facilitating comprehensive benchmarking experiments on the million song dataset. In *Proceedings of the 13th International Conference on Music Information Retrieval (ISMIR)*, 2012. ISBN 978-972-752-144-9. URL `http://dblp.uni-trier.de/db/conf/ismir/ismir2012.html#RauberSM12`.

[118] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. *arXiv preprint arXiv:1403.6382*, 2014.

[119] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011. ISBN 978-0-387-85819-7.

[120] Ruslan Salakhutdinov and Geoffrey Hinton. Deep Boltzmann Machines. 2009. URL `http://jmlr.csail.mit.edu/proceedings/papers/v5/salakhutdinov09a/salakhutdinov09a.pdf`.

[121] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

[122] J. Salamon and J. P. Bello. Feature learning with deep scattering for urban sound analysis. In *European Signal Processing Conference (EUSIPCO)*, August 2015.

[123] Kevin Schawinski, Chris Lintott, Daniel Thomas, Marc Sarzi, Dan Andreescu, Steven P Bamford, Sugata Kaviraj, Sadegh Khochfar, Kate Land, Phil Murray, et al. Galaxy zoo: a sample of blue early-type galaxies at low redshift. *Monthly Notices of the Royal Astronomical Society*, 396(2):818–829, 2009.

[124] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *Proceedings of the 20th International Conference on Artificial Neural Networks (ICANN)*, 2010.

[125] Jan Schlüter and Christian Osendorfer. Music Similarity Estimation with the Mean-Covariance Restricted Boltzmann Machine. In *Proceedings of the 10th International Conference on Machine Learning and Applications (ICMLA)*, 2011.

[126] Christian Schörkhuber, Anssi Klapuri, and Alois Sontacchi. Audio pitch shifting using the constant-q transform. *Journal of the Audio Engineering Society*, 61(7/8):562–572, 2013.

[127] Frank Seide, Gang Li, and Dong Yu. Conversational speech transcription using context-dependent deep neural networks. In *INTERSPEECH*, pages 437–440, 2011.

[128] Lior Shamir. Automatic morphological classification of galaxy images. *Monthly Notices of the Royal Astronomical Society*, 399(3):1367–1372, 2009.

[129] Lior Shamir, Anthony Holincheck, and John Wallin. Automatic quantitative morphological analysis of interacting galaxies. *Astronomy and Computing*, 2:67–73, 2013.

[130] Laurent Sifre and Stéphane Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1233–1240. IEEE, 2013.

[131] B. D. Simmons, C. Lintott, K. Schawinski, E. C. Moran, A. Han, S. Kaviraj, K. L. Masters, C. M. Urry, K. W. Willett, S. P. Bamford, and R. C. Nichol. Galaxy Zoo: bulgeless galaxies with growing black holes. *MNRAS*, 429:2199–2211, March 2013. doi: 10.1093/mnras/sts491.

[132] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[133] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013. URL `http://arxiv.org/abs/1312.6034`.

[134] Ramin A Skibba, Steven P Bamford, Robert C Nichol, Chris J Lintott, Dan Andreescu, Edward M Edmondson, Phil Murray, M Jordan Raddick, Kevin Schawinski, Anze Slosar, Alexander S. Szalay, Daniel Thomas, and Jan Vandenberg. Galaxy zoo: disentangling the environmental dependence of morphology and colour. *Monthly Notices of the Royal Astronomical Society*, 399(2):966–982, 2009.

[135] M. Slaney. Web-scale multimedia analysis: Does content matter? *MultiMedia, IEEE*, 18(2):12–15, 2011. ISSN 1070-986X. doi: 10.1109/MMUL.2011.34.

[136] Malcolm Slaney, Kilian Q. Weinberger, and William White. Learning a metric for music similarity. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, 2008.

[137] Jasper Snoek, Hugo Larochelle, and Ryan Adams. Practical bayesian optimization of machine learning algorithms. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2960–2968. 2012. URL `http://books.nips.cc/papers/files/nips25/NIPS2012_1338.pdf`.

[138] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

[139] Richard Stenzel and Thomas Kamps. Improving Content-Based Similarity Measures by Training a Collaborative Model. pages 264–271, London, UK, September 2005. University of London. URL `proceedings/1081.pdf`.

[140] S. S. Stevens, Je, and E. B. Newman. A scale for the measurement of the psychological magnitude of pitch. *J. Acoust Soc Amer*, 8:185–190, 1937.

[141] MC Storrie-Lombardi, O Lahav, L Sodre, and LJ Storrie-Lombardi. Morphological classification of galaxies by artificial neural networks. *Monthly Notices of the Royal Astronomical Society*, 259(1):8P–12P, 1992.

[142] Dan Stowell and Mark D Plumbley. Audio-only bird classification using unsupervised feature learning. In *CEUR Workshop Proceedings*, volume 1180, pages 673–684, 2014.

[143] Bob L Sturm. An analysis of the gtzan music genre dataset. In *Proceedings of the second international ACM workshop on Music information retrieval with user-centered and multimodal strategies*, pages 7–12. ACM, 2012.

[144] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1139–1147, 2013.

[145] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[146] Yichuan Tang and Abdel rahman Mohamed. Multiresolution deep belief networks. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

[147] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10: 293–302, 2002. ISSN 1063-6676.

[148] George Tzanetakis, Georg Essl, and Perry Cook. Audio analysis using the discrete wavelet transform. In *Proc. Conf. in Acoustics and Music Theory Applications*, 2001.

[149] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems 26*, 2013.

[150] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[151] Stéfan van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: Image processing in python. Technical report, PeerJ PrePrints, 2014.

[152] Chong Wang and David M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011. ISBN 978-1-4503-0813-7. doi: 10.1145/2020408.2020480. URL `http://doi.acm.org/10.1145/2020408.2020480`.

[153] Xinxi Wang and Ye Wang. Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the ACM International Conference on Multimedia*, pages 627–636. ACM, 2014.

[154] Jason Weston, Samy Bengio, and Philippe Hamel. Large-scale music annotation and retrieval: Learning to rank in joint semantic spaces. *Journal of New Music Research*, 2011.

[155] Jason Weston, Chong Wang, Ron Weiss, and Adam Berenzweig. Latent collaborative retrieval. In *Proceedings of the 29th international conference on Machine learning*, 2012.

[156] K. W. Willett, Kevin Schawinski, Brooke D. Simmons, Karen L. Masters, Ramin A. Skibba, Sugata Kaviraj, Thomas Melvin, O. Ivy Wong, Robert C. Nichol, Edmond Cheung, Chris J. Lintott, and Lucy Fortson. Galaxy zoo: the dependence of the star formation-stellar mass relation on spiral disk morphology. Feb 2015.

[157] Kyle W Willett, Chris J Lintott, Steven P Bamford, Karen L Masters, Brooke D Simmons, Kevin RV Casteels, Edward M Edmondson, Lucy F Fortson, Sugata Kaviraj, William C Keel, et al. Galaxy zoo 2: detailed morphological classifications for 304 122 galaxies from the sloan digital sky survey. *Monthly Notices of the Royal Astronomical Society*, 435(4):2835–2860, 2013.

[158] J. Wülfing and M. Riedmiller. Unsupervised learning of local features for music classification. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, 2012.

[159] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

[160] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014.

[161] Udo Zölzer. *DAFX: digital audio effects*. John Wiley & Sons, 2011.