Deep Neural Networks for Music Tagging

by

Keunwoo Choi

A thesis submitted to the University of London for the degree of
Doctor of Philosophy

Department of Electronic Engineering and Computer Science
Queen Mary University of London
United Kingdom

April 2018

Abstract

In this thesis, I present my hypothesis, experiment results, and discussion that are related to various aspects of deep neural networks for music tagging.

Music tagging is a task to automatically predict the suitable semantic label when music is provided. Generally speaking, the input of music tagging systems can be any entity that constitutes music, e.g., audio content, lyrics, or metadata, but only the audio content is considered in this thesis. My hypothesis is that we can find effective deep learning practices for the task of music tagging task that improves the classification performance.

As a computational model to realise a music tagging system, I use deep neural networks. Combined with the research problem, the scope of this thesis is the understanding, interpretation, optimisation, and application of deep neural networks in the context of music tagging systems.

The ultimate goal of this thesis is to provide insight that can help to improve deep learning-based music tagging systems. There are many smaller goals in this regard. Since using deep neural networks is a data-driven approach, it is crucial to understand the dataset. Selecting and designing a better architecture is the next topic to discuss. Since the tagging is done with audio input, preprocessing the audio signal becomes one of the important research topics. After building (or training) a music tagging system, finding a suitable way to re-use it for other music information retrieval tasks is a compelling topic, in addition to interpreting the trained system.

The evidence presented in the thesis supports that deep neural networks are powerful and credible methods for building a music tagging system.

List of Abbreviations

AUC-ROC	Area Under Curve - Receiver Operating Characteristic
CNN, Convnet	(Deep) Convolutional Neural Network
CQT	Constant-Q Transform
CRNN	Convolutional Recurrent Neural Network
DNN	Deep Neural Network
LSTM	Long Short-term Memory
LVS	Label Vector Similarity
MAE	Mean Average Error
MFCC	Mel-Frequency Cepstral Coefficient
MIR	Music Information Retrieval
MLP	Multi-Layer Perceptron
MSD	Million Song Dataset
MSE	Mean Squared Error
NCO	Normalised Co-occurrence Matrix
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
STFT	Short-Time Fourier Transform
SVM	Support Vector Machine

Chapter 1

Introduction

1.1 Motivation, hypothesis, and research questions

Motivation

The task that I have been interested in during my PhD is music tagging. Music tagging, also often called music auto-tagging, is a music information retrieval (MIR) task to automatically predict relevant tags when the audio content of music is given. The tags are often applied to a track or a short clip of music item, making the task a track-level music classification. Thanks to the diversity of music tags – which often include music genres (e.g., rock, jazz), moods (e.g., sad, happy), and instruments (e.g., guitar, vocal) – music tagging can be thought as a super-problem of genre, mood, and instrument classification. Tagging can be useful for music discovery and recommendation [1] which are one of the most commercially successful topics in MIR.

Deep learning refers to various types of machine learning algorithms where features are trained in a multiple-layer network to achieve the goal of a task. They are distinguished from 'conventional' machine learning methods with feature design approach. In the latter approach, researchers need to spend time on designing potentially useful fea-

tures for each task. Moreover, the designed features are not necessarily optimal. Deep learning approaches provides an alternative way that addresses these problems, and as a result, significantly outperformed the conventional approaches in computer vision [2] and speech processing [3].

In late 2014, when I started my PhD programme, deep learning approach for music tagging was still in its early stage. There were some papers learning to predict music tags with deep neural network [4]. However, the adoption was slow, leaving many unanswered research questions such as finding the optimal network structure, convolutional kernel sizes, and input audio conditioning strategy. Solving those questions were clearly of interest to academia, industry, and myself.

Hypothesis

In the intersection of music tagging and deep learning, I present the fundamental hypothesis.

- *We find effective deep learning practices for the task of music tagging that improves the classification performance*

To elaborate, "effective" indicates better performance and higher data/memory/computation efficiency and "practices" indicates the way to design network structure and use data.

Research questions

There are many detailed research topics to examine the hypothesis. The topics that I have been working on are summarised as below with the corresponding chapters.

- How is the music tagging dataset constructed? How accurate are they? These are

crucial when the algorithm is heavily relying on learning from the given dataset, which is true in deep learning methods (Chapter 3)

- Given the properties of music signals and tags, what kinds of new structure can we use to improve the performance? Proposing a new structure is one of the most common research topics in deep learning, requiring to exploit domain knowledge (Chapter 4)

- What are the most suitable data preprocessing methods for deep learning in music tagging and why? This is important, particularly from an engineering perspective (Chapter 5)

- After training a network for music tagging, how can we re-use the network for other MIR tasks? (Chapter 6)

- What does a convolutional neural network learn when it is trained for music classification? (Chapter 7)

This thesis presents the research I have performed to answer these questions. The following section elaborates each chapter and related papers.

1.2 Contributions and outline

I have been actively publishing papers and software. In this section, I focus on presenting the contributions that are related to each chapter of this thesis. A more comprehensive list of my works is presented in Section 1.3.

In Chapter 2, I present the backgrounds on music tagging, machine learning, and deep learning. Some of this content overlaps with my journal submission, "A Tutorial on Deep Learning for Music Information Retrieval" [5].

In Chapter 3, I extensively studied the effects of noisy labels in music tagging. It was

well known that there is a large amount of noise in music tagging datasets [6]. However, to my best knowledge, my work is the first attempt to quantify the amount of noise as well as its effects. This chapter includes why there is a large amount of noise in music tagging datasets, how much they are, and how much they affect the training and evaluation of music tagging algorithms. Additionally, I defined 'tagability' to explain the observation. This work is closely related to [7], "The Effects of Noisy Labels on Deep Convolutional Neural Networks for Music Classification" that was published in IEEE Transaction on the Emerging Topics of Computational Intelligence, 2018.

In Chapter 4, I suggest a novel network structure and compare it with three different convolutional neural network structures that have been proposed for music tagging and classifications. Because there are various aspects of network structures, only presenting the best performances does not comprehensively reflect the attributes of the structures. Instead, I scale the networks by changing the width of the network while keeping the overall structure. The performance comparison is twofold, *i)* with respect to the number of parameters which is related to the memory usage and *ii)* with respect to training time which is related to the computation complexity. This chapter is related to [8], "Convolutional Recurrent Neural Networks for Music Classification" that was presented in International Conference on Acoustics, Speech, and Signal Processing, 2017.

In Chapter 5, I discuss various audio preprocessing methods in the context of deep neural networks for music tagging. Although they affect the performance, audio preprocessing methods have not gained much attention. One of the reasons would be because comparing the performance with changing audio processing parameters (e.g., the FFT size) requires substantial time and storage. I addressed this problem by using a custom Python package, *Kapre* [9], which does real-time audio preprocessing on GPU and is developed by myself. Using Kapre, I ran a set of experiment with varying audio preprocessing methods in order to compare their effects on the performances. This chapter is linked to [10], "A Comparison on Audio Signal Preprocessing Methods for Deep Neural Networks on Music Tagging" that suggests a practical advice on how to preprocess the

audio for deep learning-based methods and was submitted to EUSIPCO 2018.

In deep learning, transfer learning is considered to be important in general. It could be even more in MIR where large-scale datasets are not always available. In Chapter 6, I introduce a novel transfer learning approach, where I show how can we transfer the knowledge that is obtained in the network by training a music tagger to other MIR problems. This is related to [11], "Transfer learning for music classification and regression tasks" which was presented in International Society of Music Information Retrieval conference, 2017 and received the best paper award.

Finally, one of the important research areas in deep learning is to understand and explain the networks. Chapter 7 includes two approaches for those purposes in music classification. First, I analyse a convnet that is trained for music classification by a technique called *auralisation*. I suggest to sonify the trained feature to listen to what each convolutional kernel focuses on so that we can understand the mechanism of convnet by listening. Second, I present an analysis of a trained network using *label vector*, which shows the inter-label similarities computed by the network. These are related to my preprints "Explaining Deep Convolutional Neural Networks on Music Classification" [12], "Auralisation of Deep Convolutional Neural Networks: Listening to Learned Features" [13], and a journal article, "The effects of noisy labels on the Deep Convolutional Neural Networks for Music Classification" [7].

I conclude my work in Chapter 8, summarising the meanings of the presented works and suggesting the direction of future work.

1.3 Publications

This section introduces the publications during my PhD. Papers are accessible via `my Google scholar page`.[1]

[1] Visit `https://scholar.google.co.kr/citations?hl=en&user=ZrqdSu4AAAAJ&view_op=list_works&sortby=pubdate` or search "Keunwoo Choi"

1.3.1 Journal papers

1. The Effects of Noisy Labels on the Deep Convolutional Neural Networks for Music Classification, Keunwoo Choi et al., *IEEE Transactions on Emerging Topics on Computational Intelligence*, 2017

2. Deep Learning for Audio-based Music Classification, Juhan Nam, Keunwoo Choi, et al., *Under review (IEEE Signal Processing Magazine)*

1.3.2 Conference papers (peer-reviewed)

1. Text-based LSTM networks for automatic music composition, Keunwoo Choi et al., *Conference on Computer Simulation of Musical Creativity*, Huddersfield, UK, 2016

2. Automatic Tagging using Deep Convolutional Neural Networks, Keunwoo Choi et al., *17th International Society for Music Information Retrieval Conference*, New York, USA, 2016

3. Convolutional Recurrent Neural Networks for Music Classification, Keunwoo Choi et al., *42nd IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, New Orleans, USA, 2017

4. Transfer Learning for Music Classification and Regression Tasks, Keunwoo Choi et al., *The 18th International Society for Music Information Retrieval Conference*, Suzhou, China, 2017

5. Similarity measures for vocal-based drum sample retrieval using deep convolutional auto-encoders, Adib Mehrabi, Keunwoo Choi, et al., *43rd IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Calgary, Canada, 2018

6. A Comparison of Audio Signal Preprocessing Methods for Deep Neural Networks on Music Tagging, Keunwoo Choi et al., *EUSIPCO, Rome, Italy, 2018*

7. Revisiting Singing Voice Detection: A quantitative review and the future outlook, Kyungyun Lee, Keunwoo Choi, et al., *The 19th International Society for Music Information Retrieval Conference*, Paris, France, 2018

1.3.3 Preprints, workshop papers, and abstracts

1. Understanding Music Playlists, Keunwoo Choi et al., *Machine Learning for Music Discovery Workshop at 32nd International Conference on Machine Learning*, Lille, France, 2015

2. Auralisation of Deep Convolutional Neural Networks: Listening to Learned Features, Keunwoo Choi et al., *Late-Breaking Demo Session of 16th International Society of Music Information Retrieval Conference*, Malaga, Spain, 2015

3. Towards Playlist Generation Algorithms using RNNs Trained on Within-Track Transitions, Keunwoo Choi et al., *SOAP Workshop (Workshop on Surprise, Opposition, and Obstruction in Adaptive and Personalized Systems)*, Halifax, NS, Canada, 2016

4. Explaining Deep Convolutional Neural Networks on Music Classification, Keunwoo Choi et al., *arXiv:1607.02444*, 2016

5. Towards Music Captioning: Generating Music Playlist Descriptions, Keunwoo Choi et al., *Late-Breaking/Demo session of 17th International Society of Music Information Retrieval Conference*, New York, USA, 2016

6. Kapre: On-GPU Audio Preprocessing Layers for a Quick Implementation of Deep Neural Network Models with Keras, Keunwoo Choi et al., *Machine Learning for Music Discovery Workshop at 32nd International Conference on Machine Learn-*

ing, Sydney, Australia, 2017

7. A Tutorial on Deep Learning for Music Information Retrieval, Keunwoo Choi et al., *Under review (Transaction of ISMIR)*

1.4 Software

List of the software I released during my PhD.

1. Kapre: Keras Audio Preprocessor

   `https://github.com/keunwoochoi/kapre`

   Kapre is a Python package for faster implementation of audio signal preprocessing in a deep learning workflow. It was released in December 2016 and has been starred for more than 200 times on Github and currently used by many researchers.

2. LSTM Realbook

   `https://github.com/keunwoochoi/lstm_real_book`

   This repository consists of the dataset and RNN models for my paper, "Text-based LSTM networks for automatic music composition" [14] (CSMC 2016).

3. Convnet-based music taggers

   `https://github.com/keunwoochoi/music-auto_tagging-keras`

   This repository includes the trained music tagger in my paper, "Automatic Tagging Using Deep Convolutional Neural Networks" [15] (ISMIR 2016).

4. Music tagger-based feature extractor

   `https://github.com/keunwoochoi/transfer_learning_music`

   This repository consists of a trained music feature extractor in my paper, "Transfer learning for music classification and regression tasks" [11] (ISMIR 2017).

5. dl4mir - deep learning for music information retrieval

   `https://github.com/keunwoochoi/dl4mir`

dl4mir consists of a series of educational Python codes for deep learning beginners in MIR research.

Chapter 2

Background

Abstract

This chapter introduces the music tagging problem and basic concepts of machine learning and deep learning. A precise definition and detailed explanation of music tagging are provided to help the reader to understand this thesis. The following subsections for machine learning and deep learning can be considered as a short tutorial on the field, as well as introductory sections for the terms of this thesis. Finally, I introduce Compact-convnet, a simple 5-layer convolutional neural network that I use for several times throughout this thesis.

The machine learning and deep learning sections are closely related to my tutorial paper, "A tutorial on deep learning for music information retrieval" [5].

2.1 Music tagging

2.1.1 Music tags

Music tags are descriptive keywords that convey various types of high-level information about recordings such as mood ('sad', 'angry', 'happy'), genre ('jazz', 'classical') and instrumentation ('guitar', 'strings', 'vocal', 'instrumental') [15]. Tags may be associated with music in the context of a folksonomy, i.e., user-defined metadata collections commonly used for instance in online streaming services, as well as personal music collection management tools. As opposed to expert annotation, these types of tags are deeply related to listeners' or communities' subjective perception of music. In the aforementioned tools and services, a range of activities including search, navigation, and recommendation may depend on the existence of tags associated with tracks. New and rarely accessed tracks however often lack the tags necessary to support them, which leads to well-known problems in music information management [6]. For instance, tracks or artists residing in the long tail of popularity distributions associated with large music catalogues may have insufficient tags, therefore they are rarely recommended or accessed

and tagged in online communities. This leads to a circular problem. Expert annotation is notoriously expensive and intractable for large catalogues, therefore content-based annotation is highly valuable to bootstrap these systems. Music tag prediction is often called *music auto-tagging* [1]. Content-based music tagging algorithms aim to automate this task by learning the relationship between tags and the audio content.

Music tagging can be seen as a multi-label classification problem because music can be correctly associated with more than one true label, for example, {'rock', 'guitar', 'happy', and '90s'}. This example also highlights the fact that music tagging may be seen as multiple distinct tasks from the perspectives of machine learning and music informatics. Because tags may be related to genres, instrumentation, mood and era, the problem may be seen as a combination of genre classification, instrument recognition, mood and era detection, and possibly others. In the following, I highlight three aspects of the task that emphasise its importance in music information retrieval (MIR).

First, collaboratively created tags reveal significant information about music consumption habits. Tag counts show how listeners label music in the real-world, which is often very different from the decision of a limited number of experts (see Section 3.2.1) [16]. The first study on automatic music tagging proposed the use of tags to enhance music recommendation [1] for this particular reason. Second, the diversity of tags and the size of tag datasets make them relevant to several MIR problems including genre classification and instrument recognition. In the context of deep learning, tags can particularly be considered a good *source task* for transfer learning [11, 17], a method of reusing a trained neural network in a related task, after adapting the network to a smaller and more specific dataset. Since a music tagger can extract features that are relevant to different aspects of music, tasks with insufficient training data may benefit from this approach. Finally, investigating trained music tagging systems may contribute to our understanding of music perception and music itself. For example, analysing subjective tags such as mood and related adjectives can help building computational models for human perception of music.

The importance of music tagging is clear both from the perspectives of recommendation and broader music informatics systems, as well as research including human perception and behaviour, yet there are several issues one faces when analysing music tags. A severe problem, particularly in the context of deep learning is the fact that sufficiently large training datasets are only available in the form of folksonomies. In these user-generated metadata collections, tags not only describe the content of the annotated items, for instance, well-defined categories such as instruments that appear on a track or the release year of a record, but also subjective qualities and personal opinions about the items [18]. Tags are often related to organisational aspects, such as self-references and personal tasks [19]. For instance, users of certain music streaming services frequently inject unique tags that have no significance to other users, i.e., they label music with apparently random character sequences which facilitate the creation of virtual personal collections, misappropriating this feature of the service. While tags of this nature are relatively easy to recognise and disregard using heuristics, other problems of folksonomies are not easily solved and constitute a great proportion of noise in these collections. Relevant problems include mislabelling, the use of highly subjective tags, such as those pertaining to genre or mood, as well as heterogeneity in the taxonomical organisation of tags. Researchers have been proposing to solve these problems either by imposing pre-defined classification systems on social tags [18], or providing tag recommendation based on context to reduce tagging noise in the first place [20]. While the benefits of such organisation or explicit knowledge of tag categories have been shown to benefit automatic music tagging systems e.g. in [21], most available large folksonomies still consist of noisy labels.

2.1.2 Music tagging as a machine learning problem

Music tagging is related to common music classification and regression problems such as genre classification and emotion prediction. The majority of prior research has focussed on extracting relevant music features and applying a conventional classifier or regres-

sor. For example, the first auto-tagging algorithm [1] proposed the use of mid-level audio descriptors such as mel-frequency cepstral coefficients (MFCCs) and an AdaBoost [22] classifier. Since most audio features are extracted frame-wise, statistical aggregates such as mean, variance and percentiles are also commonly used. This is based on the assumption that the features adhere to a pre-defined or known distribution which may be characterised by these parameters. However, hand-crafted audio features do not necessarily obey known parametric distributions [23, 24]. Consequently, vector quantisation and clustering were proposed e.g. in [25] as an alternative to parametric representations.

A recent trend in music tagging is the use of data-driven methods to *learn* features instead of designing them, together with non-linear mappings to more compact representations relevant to the task. These approaches are often called *representation learning* or *deep learning*, due to the use of multiple layers in neural networks that aim to learn both low-level features and higher-level semantic categories. Convolutional Neural Networks (denoted 'convnets' hereafter) have been providing state-of-the-art performance for music tagging in recent works [4], [15], [17].

2.1.3 Evaluation of music tagging algorithms

There are several methods to evaluate tagging algorithms. Since the target is typically binarised to represent if the i^{th} tag is true or false ($y_i \in \{0, 1\}$), classification evaluation metrics such as 'Precision' and 'Recall' can be used if the prediction is also binarised. Because label noise is mostly associated with negative labels, as quantified in Section 3.2.2, using recall is appropriate since it ignores incorrect negative labels. They can be used instead as an auxiliary method of assessment after training. This strategy can work well because it prevents the network from learning trivial solutions for those metrics. For instance, predicting all labels to be 'True' to obtain a perfect recall score. Optimal thresholding for binarised prediction is an additional challenge however and discards information. The network learns a maximum likelihood solution with respect to the training data, which is heavily imbalanced, therefore the threshold should be chosen

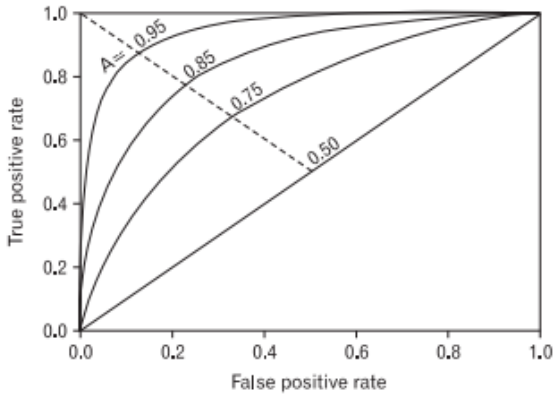

Figure 2.1: An example of AUC-ROC curves. Please note that there are infinite numbers of the curves that exhibits an equal AUC-ROC score.

specifically for each tag. This introduces an entirely new research problem which I do not address here.

The area under curve - receiver operating characteristic (AUC-ROC, or simply AUC) works without binarisation of predictions and is often used as an evaluation metric. A ROC curve is created by plotting the true positive rate against the false positive rate. As both rates range between $[0, 1]$, the area under the curve also ranges between $[0, 1]$. However, the effective range of AUC is $[0.5, 1]$ since random classification yields 0.5 when the true positive rate increases at the exact same rate of false positives. Figure 2.1 illustrates examples of AUC curves when scores are 0.95, 0.85, 0.75, and 0.50.

2.2 Machine learning

It is not the scope of this thesis to provide a comprehensive overview of machine learning. This section only covers the fundamentals of machine learning that would help to understand the motivation of deep learning.

2.2.1 General concepts

Data-driven approaches One of the aspects that defines machine learning from other mathematical and/or computational approaches is that it *learns* how to capture the relevant patterns from the provided data (example). It is shown in the name itself, 'machine learning', which indicates the procedure that *lets machine learn from data*.

Training indicates the procedure of machine learning something using given data which is called as training data.

Supervised learning There are several categories of algorithms and approaches under machine learning. Let me first describe supervised learning since it is the most relevant category to the thesis. Supervised learning is a type of machine learning approaches where the training data consists of a set of input $\mathbb{X}$ and another set of output (or label), $\mathbb{Y}$. The machine is trained to learn the relationship between inputs and outputs.

This is in contrast to other categories. For example, during *unsupervised learning*, only the input data is provided. Principal component analysis, clustering algorithms such as K-means clustering, and non-negative matrix factorisation are of this type.

Classification and regression problems Supervised learning algorithms usually perform either classification or regression. Classification is a process that predicts a qualitative (or categorical) label for an input. For example, in MIR, genre classification task is to predict the genre label(s) of music input.

On the other hand, regression is a process that predicts a quantitative response for an input. For example, in MIR, music emotion prediction is often formulated as a regression problem where the algorithm predicts the levels of 'arousal' and 'valence' for given music input. The response may be bounded in a range, e.g., arousal and valence are often bounded in $[0, 1]$.

There can be an ambiguity between classification and regression. For example, on one hand, a classification algorithm can be based on a regression which predicts the probabilities for each class. On the other hands, regression problems are sometimes converted into a classification problem by quantising (or binning) the prediction range.

Feature extractor and classifier/regressor In many cases, machine learning methods consist of two stages; a feature extractor and a classifier/regressor.

A feature extractor takes the raw input data (e.g., audio signals, images, documents) and extracts a representation that is (hoped to be) useful for the prediction of given task. Usually the feature is expected to be in a smaller dimension than the original data so that the following procedure can be performed efficiently. For audio signal, MFCCs are one of the most popular and robust features. Different features are designed and used in different domains and designing/selecting features require background knowledge as well as substantial efforts.

A classifier (or regressor) takes the feature as an input and makes a prediction. Usually, this is the procedure that involves learning from the training data. The classifier learns to map the features to the label using the ground truth. Popular classifiers are linear regression, support vector machine, K-nearest neighbour, random forest, XGBoost, etc.

Concepts in artificial neural networks A **node** is analogous to a biological neuron and represents a scalar value as in Figure 2.2 (a). A **layer** consists of a set of nodes as in Figure 2.2 (b) and represents a vector. Note that the nodes within a layer are usually not inter-connected. Usually, an **activation function** $f()$ is applied to each node as in Figure 2.2 (c). A node may be considered to be activated/deactivated by the output value of an activation function. The value of a node is usually computed as a weighted sum of the input followed by an activation function, i.e., $f(w \cdot x)$ where w is the weights and x is the inputs as in Figure 2.2 (d). A simple artificial neural network is illustrated

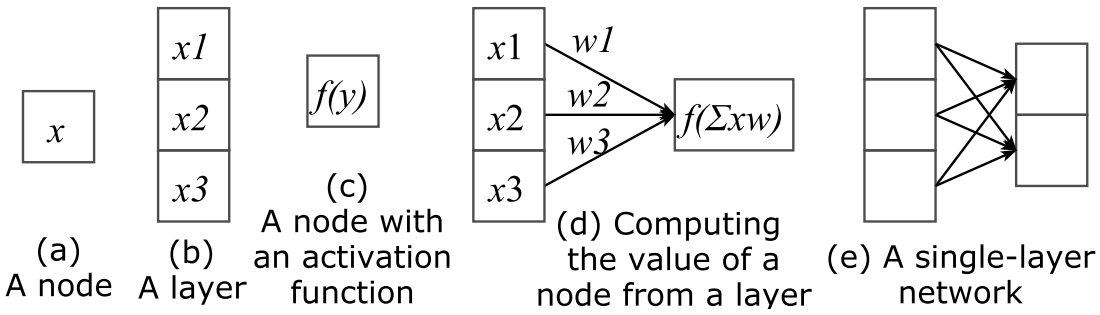

Figure 2.2: Illustrations of (a) a node and (b) a layer. A node often has an nonlinear function called activation function $f()$ as in (c). As in (d), the value of a node is computed using the previous layer, weights, and an activation function. A single-layer artificial neural network in (e) is an ensemble of (d).

in Figure 2.2 (e), which is an extension of Figure 2.2 (d). In a network with multiple layers, there are intermediate layers as well as the input and the output layer which are also called **hidden layers**. The **depth** of a network is the total number of layers in a network, often ignoring the input layer. This is because the input layer itself does not have parameters (or weights) that do any computation. Similarly, the **width**(s) of a network is the number of nodes in layer(s).

When referring the layers of a network, 'earlier layers' means the layers that are closer to the input, while 'deeper layers' means the layers that are closer to the output.

2.3 Deep learning

In recent years, deep learning methods have become more popular in the field of MIR research. For example, while there were only 2 deep learning articles in 2010 in ISMIR conferences [1] ([26], [27]) and 6 articles in 2015 ([28], [29], [30], [31], [32], [33]), it increases to 16 articles in 2016. This trend is even stronger in other machine learning fields, e.g., computer vision and natural language processing, those with larger communities and more competition. Overall, deep learning methods are probably going to serve an

[1] `http://www.ismir.net`

essential role in MIR.

There were a number of important early works on neural networks that are related to the current deep learning technologies. The error backpropagation [34], which is a way to apply the gradient descent algorithm for deep neural networks, was introduced in the 80s. A convolutional neural network (convnet) was used for handwritten digit recognition in [35]. Later, long short-term memory (LSTM) recurrent unit was introduced [36] for sequence modelling. They still remain the foundation of modern DNN algorithms.

Recently, there have been several advancements that have contributed to the success of the modern deep learning. The most important innovation happened in the optimisation technique. The training speed of DNNs was significantly improved by using rectified linear units (ReLUs) instead of sigmoid functions [37] (Figure 2.4). This led to innovations in image recognition [2] and speech recognition [38] [39]. Another important change is the advance in hardware. Parallel computing on graphics processing units (GPUs) enabled Krizhevsky et al. to pioneer a large-scale visual image classification in [2].

'Conventional' machine learning approaches involve hand-designing features and having the machine learn a classifier as illustrated in Figure 2.3 (a). For example, one can use MFCCs, assuming they provide relevant information for the task, then train a classifier (e.g., logistic regression), that maps the MFCCs to the label. Therefore, only a part of the whole procedure (e.g., classifier) is learned while the other (e.g., computing MFCCs) is not data-dependent.

On the contrary, deep learning approaches assume multiple trainable layers, all of which learn from the data, as in Figure 2.3 (b). Having multiple layers is important because when they are combined with nonlinear activation functions, a network learns complicated relationships between the input and the output. Because of this fully trainable connection, deep learning is also called end-to-end learning. For example, the input and output can be audio signals and genre labels respectively.

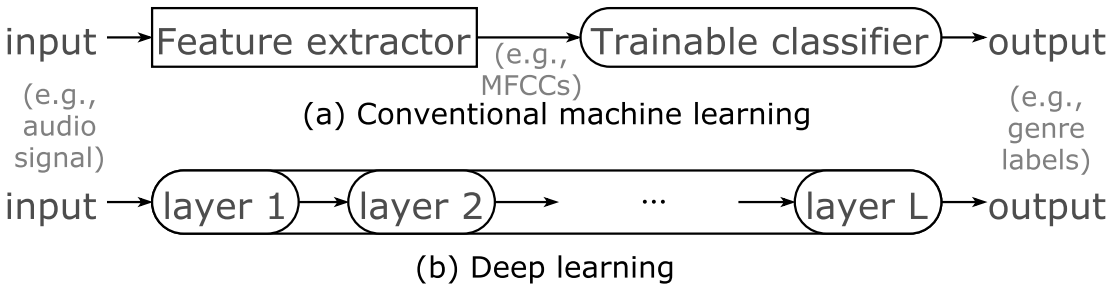

Figure 2.3: Block diagrams of conventional machine learning and deep learning approaches. Trainable modules are in rounded rectangular.

2.3.1 Designing and training neural networks

Designing a neural network structure involves selecting types and numbers of layers and loss function relevant to the problem. These parameters, that govern the network architecture, are called hyperparameters. Let's consider neural networks as function approximators $f : X \to Y$ with given input X and output Y in the data. A network structure constrains the function form; and during designing a network, the constraints are compromised to minimise redundancy in the flexibility while maximising the capacity of the network.

After the design process, a network is parametrised by its weights w. In other words, an output of a network $\hat{y}$ is a function of input x and the weights w. A **loss function** $J(w)$ is used to measure the difference between the predicted output $\hat{y}$ and the ground truth output y with respect to the current weights w. A loss function is decided so that minimising it would lead to achieving the goal of the task.

Training a network is an iterative process of adjusting the weights w to reduce the loss $J(w)$. A loss function provides a way to evaluate the prediction with respect to the true label. A de-facto optimisation method is the gradient descent which iteratively updates the parameters in such a way that the loss decreases most rapidly, as formulated in Eq. 2.1.

$$w := w - \eta \nabla J(w) \tag{2.1}$$

where η is the learning rate and $\nabla J(w)$ is the gradient of J(w). As mentioned earlier, in DNNs, gradient descent over multiple layers is called backpropagation [34], which computes the gradient of loss function with respect to the nodes in multiple layers using the chain rule.

The **convergence speed** may affect the overall performance because a significant difference in it may result in different convergences of the training. As in Eq. 2.1, a successful training is a matter of controlling the learning rate η and the gradient $\nabla J(w)$.

The **learning rate** η can be adaptively controlled, which is motivated by the intuition that the learning rate should decrease as the loss approaches local minima. As of 2017, Adam is one of the most popular methods [40], providing an adaptive controlling of the learning rate using the accumulated gradient per each dimension. An overview of adaptive learning rate can be found in an overview article on gradient descent algorithms [41] as well as [42].

The **gradient** $\nabla J(w)$ is crucial for training DNNs. Backpropagation, the norm of training method for neural networks, is based on the chain rule of gradients. A good *gradient flow* leads to high performance and is the fundamental idea of innovations such as LSTM unit [36], highway networks [43], and deep residual networks [44], all of which are showing state-of-the-art performances in sequence modelling and visual image recognition.

Activation functions play an important role in DNNs. Not only they are analogous to the activation of biological neurons in some sense, but also they introduce non-linearity between layers and it enables the whole network to learn more complicated patterns.

Sigmoid functions (e.g., the logistic function, $f(x) = \frac{1}{(1+e^{-x})}$ and the hyperbolic tangential function, $f(x) = \frac{(e^{x}-e^{-x})}{(e^{x}+e^{-x})}$) were used in early neural network works until early

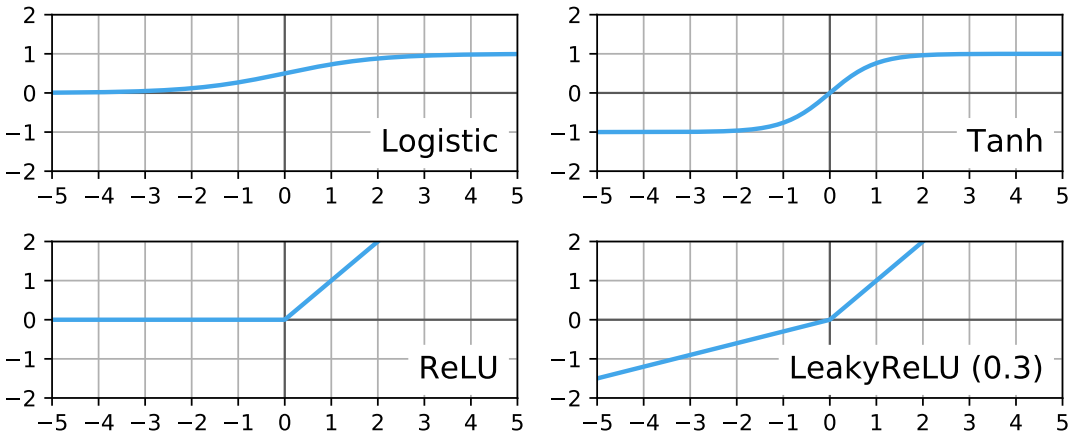

Figure 2.4: Four popular activation functions - a logistic, hyperbolic tangential, rectified linear unit (ReLU), and leaky ReLU.

2010s. However, a network with sigmoid activation functions may have the 'vanishing gradient' problem [45] which impedes training DNNs. The vanishing gradient problem happens when the gradient flow becomes too slow due to a very small gradient, $\nabla J(w)$ (see [45] for further details).

ReLU was introduced as an alternative to solve the vanishing gradient problem [37]. Its response is illustrated in 2.4. ReLU has been the first choice in many recent applications in deep learning.

For the output layer, it is recommended to use an activation function that has the same output range to the range of the ground truth. For example, if the label y is a probability, Sigmoid function would be most suitable for its output ranges in $[0, 1]$. If the task is single-label classification problem, the softmax is preferred because correct gradients can be provided from all the output nodes. If it is a regression problem with an unbounded range, a linear activation function can be used.

2.3.2 Audio data representation for deep learning

In this section, several audio data representations are reviewed in the context of using deep learning methods. Majorities of deep learning approaches in MIR take advantage of 2-dimensional representations instead of the original 1-dimensional representation which is the (discrete) audio signal. In many cases, the two dimensions are frequency and time axes.

When applying deep learning methods to MIR problems, it is particularly important to understand the properties of audio data representations. Training DNNs is computationally intensive, therefore optimisation is necessary for every stage. One of the optimisations is to pre-process the input data so that it represents its information effectively and efficiently – effectively so that the network can easily use it and efficiently so that the memory usage and/or the computation is not too heavy.

In many cases, two-dimensional representations provide audio data in an effective form. By decomposing the signals with kernels of different centre frequencies (e.g., STFT), audio signals are *separated*, i.e., the information of the signal becomes clearer.

Although those 2D representations have been considered as visual images and these approaches have been working well, there are also differences. Visual images are locally correlated; nearby pixels are likely to have similar intensities and colours. In spectrograms, there are often harmonic correlations which are spread along frequency axis while local correlation may be weaker. [46] and [47] focus on the harmonic correlation by modifying existing 2D representations, which is out of the scope of this paper but strongly recommended to read. A scale invariance is expected for visual object recognition but probably not for music/audio-related tasks.

•**Audio signal**: The time-series audio signal is often called *raw* audio, compared to other representations that are transformations based on it. A digital audio signal consists of audio samples that specify the amplitudes at time-steps. In the majority of

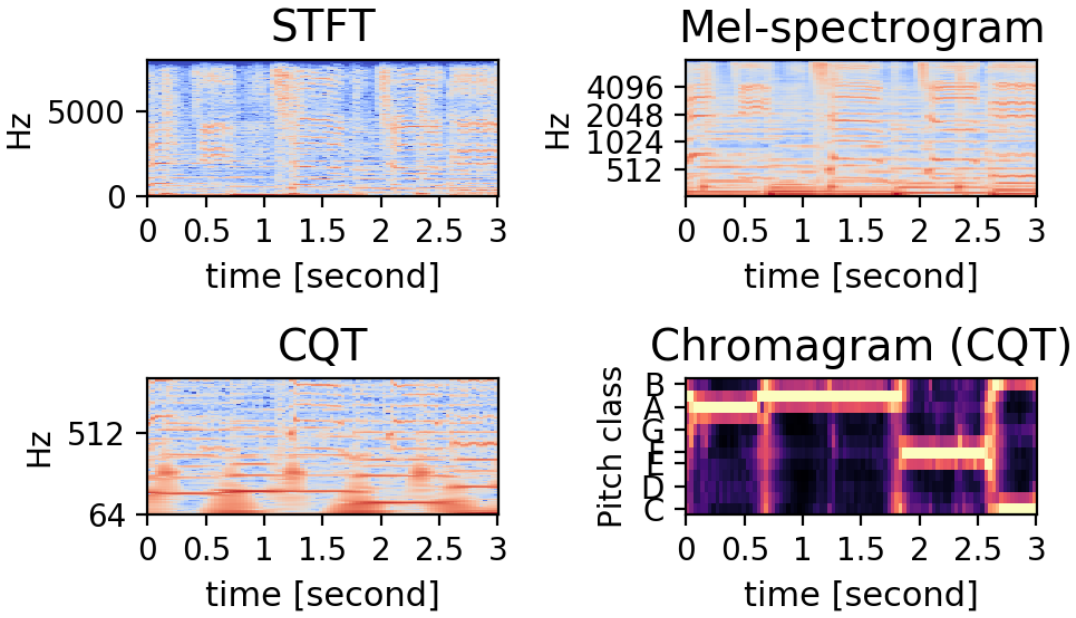

Figure 2.5: Audio content representations. STFT, mel-spectrogram, CQT, and a chromagram of a music signal are plotted. Please note the different scales of frequency axes of STFT, mel-spectrogram, and CQT.

MIR works, researchers assume that the music content is given as a digital audio signal, isolating the task from the effect of acoustic channels. The time-series audio signal has not been the most popular choice; researchers have preferred 2D representations such as STFT and mel-spectrograms because learning a network starting from the audio signal requires even a larger computation power.

Recently, however, one-dimensional convolutions are often used to learn an alternative of existing time-frequency conversions, e.g., in music tagging [4, 17].

•**Short-Time Fourier Transform**: STFT provides a time-frequency representation with linearly-spaced centre frequencies. The computation of STFT representation is usually quicker than other time-frequency representations thanks to fast Fourier transform (FFT) which reduces the cost $O(N^2)$ to $O(N \log(N))$ with respect to the number of FFT points and parallel computing.

The linear centre frequencies are not always desired in music analysis. They do not match the frequency resolution of the human auditory system, nor musically motivated like the frequencies of constant-Q transform (CQT, explained later in this subsection). This is why STFT is not the most popular choice in deep learning – it is not efficient in size as mel-spectrogram and not as raw as audio signals.

One of the merits of STFT is that it is perfectly invertible to the audio signal, for which STFT was used in sonification of learned features [12] and source separation [48].

• **Mel-spectrogram**: Mel-spectrogram is a 2D representation that is optimised for human auditory perception. It compresses the STFT in the frequency axis and therefore can be more efficient in its size while preserving the most perceptually important information. Mel-spectrogram only provides the magnitude (or energy) of the time-frequency bins and it is not invertible to time-series audio signals.

There are other scales that are similar to mel-bands and based on the psychology of hearing – the bark scale, equivalent rectangular bandwidth (ERB), and gammatone filters [49]. They have not been compared in MIR context but in speech research and the result did not show a significant difference on mel/bark/ERB in speech synthesis [50] and mel/bark for speech recognition [51].

There are many suggestions on composing mel-frequencies. [52] suggests the formula as Eq. 2.2.

$$m = 2595 \log_{10}(1 + \frac{f}{700}) \tag{2.2}$$

for frequency f in hertz. Trained kernels of end-to-end configuration have resulted in nonlinear frequencies that are similar to log-scale or mel-scale [4, 17]. Those results agree with the known human perception [49] and indicate that the mel-frequencies are quite suitable for the those tasks, perhaps because tagging is a subjective task. For those empirical and psychological reasons, mel-spectrograms have been popular for tagging

[53] [15], [8], boundary detection [54], onset detection [55] and learning latent features of music recommendation [56].

• **Constant-Q Transform** (CQT): CQT provides a 2D representation with logarithmic-scale centre frequencies. This is well matched to the frequency distribution of the pitch, hence CQT has been predominantly used where the fundamental frequencies of notes should be precisely identified, e.g. chord recognition [57] and transcription [58].The centre frequencies are computed as Eq. 2.3.

$$f_c(k_{lf}) = f_{min} \times 2^{k_{lf}/\beta}, \tag{2.3}$$

where f_{min}: minimum frequency of the analysis (Hz), k_{lf} : integer filter index, β : number of bins per octave, Z : number of octaves.

Note that the computation of a CQT is heavier than that of an STFT or mel-spectrogram. (As an alternative, log-spectrograms can be used and showed even a better performance than CQT in piano transcription [59].)

• **Chromagram** [60]: The chromagram, also often called the pitch class profile, provides the energy distribution on a set of pitch classes, often with the western music's 12 pitches.[61] [62]. One can consider a chromagram as a CQT representation folding in the frequency axis. Given a log-frequency spectrum X_{lf} (e.g., CQT), it is computed as Eq. 2.4.

$$C_f(b) = \sum_{z=0}^{Z-1} |X_{lf}(b + z\beta)|, \tag{2.4}$$

where z=integer, octave index, b=integer, pitch class index $\in [0, \beta - 1]$.

Like MFCCs, chromagram is more 'processed' than other representations and can be used as a feature by itself.

Symbols	Meaning
N	Number of channels of a 2D representation
F	Frequency-axis length of a 2D representation
T	Time-axis length of a 2D representation
H	Height of a 2D convolution kernel (frequency-axis)
W	Width of a 2D convolution kernel (time-axis)
V	Number of hidden nodes of a layer
L	Number of layers of a network

Table 2-A: Symbols and their meanings defined in this paper. Subscript indicates the layer index, e.g., N_1 denotes the number of channels (feature maps) in the first convolutional layer.

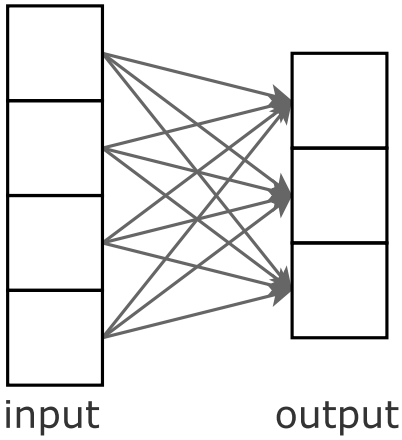

Figure 2.6: An illustration of a dense layer that has a 4D input and 3D output.

In the following sections, the frequently used layers in deep learning are introduced. For each type of a layer, a general overview is followed by a further interpretation in the MIR context. Hereafter, the symbols in layers are defined as in Table 2-A.

2.3.3 Dense layers

A dense layer is a basic module of DNNs. Dense layers have many other names - dense layer (because the connection is dense), fully-connected layers (because inputs and outputs are fully-connected), affine transform (because there is $\mathbf{W} \cdot x + b$ as in Eq. 2.5), MLP (multi-layer perceptron which is a conventional name of a neural network), and confusingly and unlike in this paper, DNNs (deep neural networks, but to denote deep

neural networks only with dense layers). A dense layer is formulated as Eq. 2.5,

$$y = f(\mathbf{W} \cdot x + b), \tag{2.5}$$

where x and $b \in \mathbb{R}^{V_{\text{in}}}$, $y \in \mathbb{R}^{V_{\text{out}}}$, $\mathbf{W} \in \mathbb{R}^{V_{\text{in}} \times V_{\text{out}}}$, and each corresponds to input, bias, output, and the weight matrix, respectively. $f()$ is a nonlinear activation function (Sec 2.3.1 for details).

The input to a dense layer with V nodes is transformed into a V-dimensional vector. In theory, a single node can represent a huge amount of information as long as the numerical resolution allows. In practice, information is often spread over more than one node. Sometimes, a narrow layer (a layer with a small V) can work as a bottleneck of the representation. For networks in many classification and regression problems, the dimension of output is smaller than that of input, and the widths of hidden layers are decided between V_{out} and V_{in}, assuming that the representations become more compressed, in higher-levels, and more relevant to the prediction in deeper layers.

2.3.3.1 Dense layers and music

In MIR, a common usage of a dense layer is to learn a frame-wise mapping as in (**d1**) and (**d2**) of Figure 2.8. Both are non-linear mappings but the input is multiple frames in **d2** in order to include contextual information around the centre frame. By stacking dense layers on the top of a spectrogram, one can expect that the network will learn how to reshape the frequency responses into vectors in another space where the problem can be solved more easily (the representations becomes *linearly separable*[2]). For example, if the task is pitch recognition, we can expect the first dense layer to be trained in such a way that its each output node represents different pitch.[3]

[2] `http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/` for a further explanation and demonstration.

[3] See example 1 on `https://github.com/keunwoochoi/dl4mir`, a simple pitch detection task with a dense layer.

By its definition, a dense layer does not facilitate a shift or scale invariance. For example, if an STFT frame length of 257 is the input of a dense layer, the layer maps vectors from 257-dimensional space to another V-dimensional space. This means that even a tiny shift in frequency, which we might hope the network be invariant to for certain tasks, is considered to be a totally different representation.

Dense layers are mainly used in early works before convnets and RNNs became popular. It was also when the learning was less of end-to-end for practical reasons (computation power and dataset size). Instead of the audio data, MFCCs were used as input in genre classification [63] and music similarity [64]. A network with dense layers was trained in [65] to estimate chroma features from log-frequency STFTs. In [66], a dense-layer network was used for source separation. Recently, dense layers are often used in hybrid structures; they were combined with Hidden Markov model for chord recognition [67] and downbeat detection [68], with recurrent layers for singing voice transcription [69], with convnet for piano transcription [59], and on cepstrum and STFT for genre classification [70].

2.3.4 Convolutional layers

The operation in convolution layers can be described as Eq. 2.6.

$$y^j = f(\sum_{k=0}^{K-1} \mathbf{W}^{jk} * x^k + b^j), \tag{2.6}$$

where all y^j, $\mathbf{W}^{jk}$, x^k, and b^j are 2-dimensional and the superscripts denote the channel indices. y^j is j-th channel output, x^k is k-th channel input, $*$ is convolution operation, $\mathbf{W}^{jk} \in \mathbb{R}^{h \times l}$ is the convolution kernel which associates k-th input channel and j-th output channel, and b^j is the bias for j-th output channel (h and l are the convolution kernel lengths in frequency and time axes). The total weights ($\mathbf{W}^{jk}$ for all j and k) are in a 4-dimensional array with a shape of (h, l, K, J) while x and y are

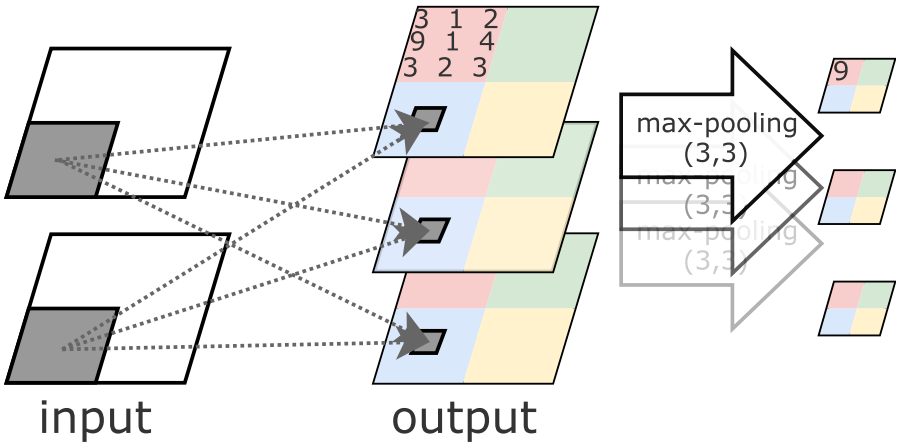

Figure 2.7: An illustration of a convolutional layer in details, where the numbers of channels of input/output are 2 and 3, respectively. The dotted arrows represent a convolution operation in the region, i.e., a dot product between convolutional kernel and local regions of input.

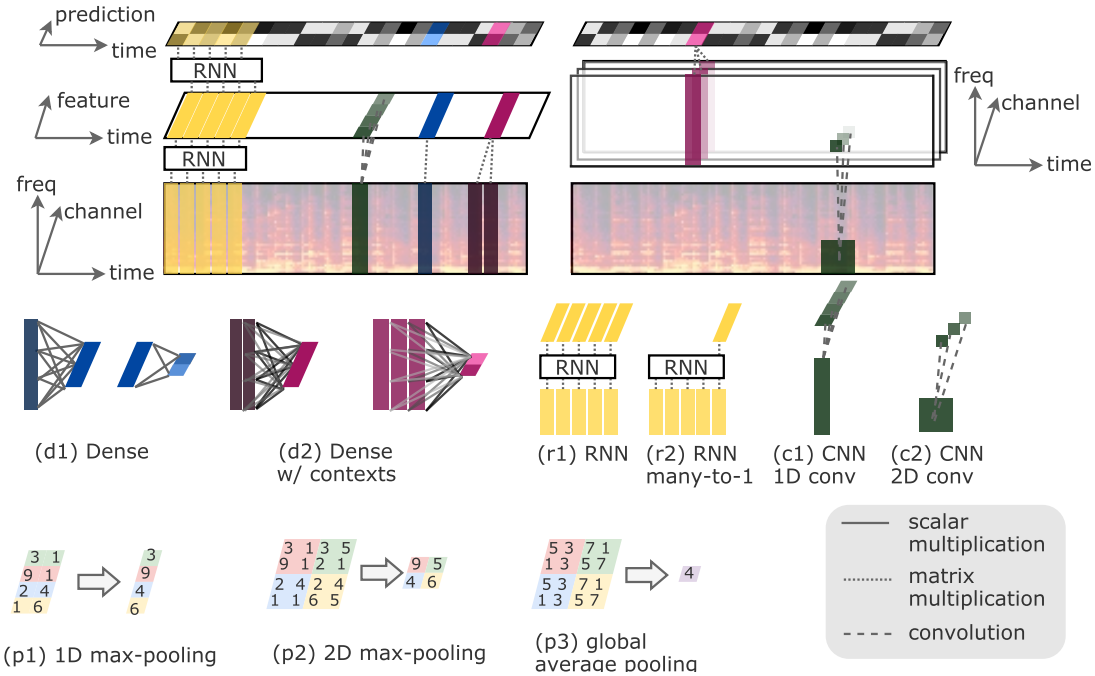

Figure 2.8: Neural network layers in the context of MIR

3-dimensional arrays including channel indices (axes for height, width, and channel). k, j are the numbers of input and output channels, and K is the number of input channels.

A 2D convolutional kernel 'sweeps' over an input and this is often described as 'weight

sharing', indicating that the same weights (convolutional kernels) are applied to the whole input area. This results in vastly reducing the number of trainable parameters.

For a given input, as a result of the sweeping, the convolutional layers output representation of local activations of patterns. This representation is called *feature map* (y in Eq. 2.6). As a result, unlike dense or recurrent layers, convolutional layers preserves the spatiality of the input.

In essence, the convolution computes a local correlation between the kernel and input. During training, the kernels learn local patterns that are useful to reduce the loss. With many layers, kernels can learn to represent some complex patterns that combine the patterns detected in the previous layers.

2.3.5 Subsampling

Convolutional layers are very often used with pooling layers. A pooling layer reduces the size of feature maps by downsampling them with an operation, usually the *max* function (Figure 2.7). Using max function assumes that on the feature maps, what really matters is if there exists an activation or not in a local region, which can be captured by the local maximum value. This non-linear subsampling provides distortion and translation invariances because it discards the precise location of the activations.

The average operation is not much used in pooling except in a special case where it is applied *globally* after the last convolutional layer [71]. This global pooling is used to summarise the feature map activation on the whole area of input, which is useful when input size varies, e.g., [72].

2.3.5.1 Convolutional layers and music

Figure 2.8 (**c1**) and (**c2**) show how 1D and 2D convolutions are applied to a spectrogram. By stacking convolutional layers, a convnet can learn more complicated patterns from

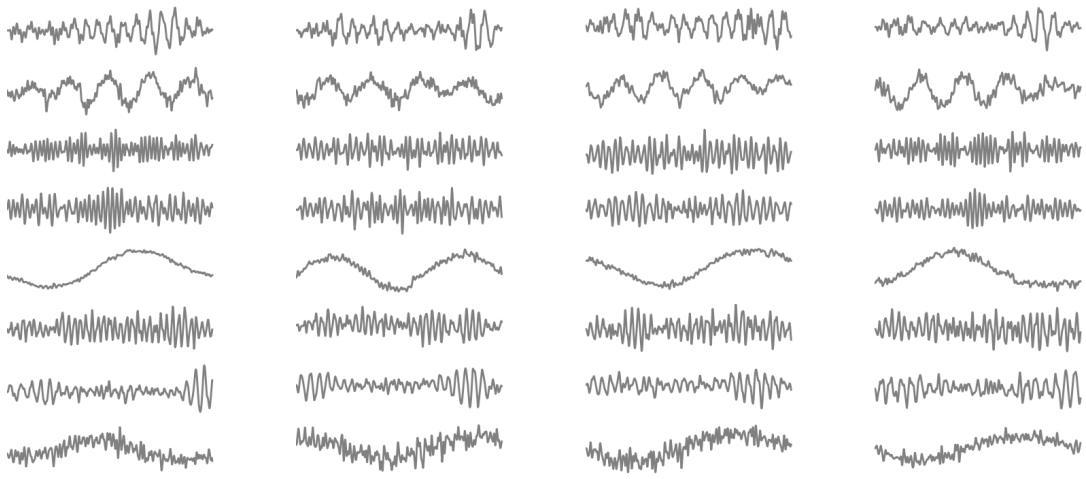

Figure 2.9: Learned 1D convolutional kernels that are applied to audio samples in [4] for music tagging

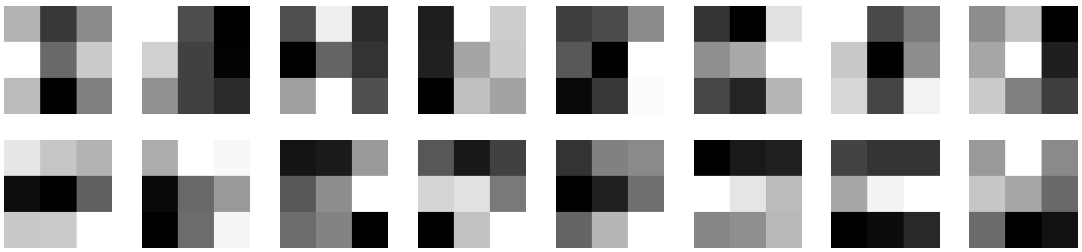

Figure 2.10: Learned 3×3 convolutional kernels at the first layer of a convnet used in [11]. These kernels are applied to log-mel-spectrograms for music tagging.

music content [12, 13].

The convolutional layer applied to the input data provides some insights of the mechanism underneath the convnet. Figure 2.9 illustrates the learned 1D convolutional kernels that are applied to raw audio samples in [4]. The kernels learned various fundamental frequencies. A more common approach is to apply 2D convolutional layers to 2D time-frequency representations. Figure 2.10 illustrates a subset of the 2D kernels that are applied to mel-spectrograms for a music tagging problem [15]. Kernels size of 3-by-3 learned vertical edge detectors (on the top row) and horizontal edge detectors (on the lower row). As in the convnets in other domains, these first-layer kernels are combined

to create more complex features.

The kernel size determines the maximum size of a component that the kernel can precisely capture in the layer. How small can a kernel be in solving MIR tasks? The layer would fail to learn a meaningful representation if the kernel is smaller than the target pattern. For example, for a chord recognition task, a kernel should be big enough to capture the difference between major and minor chords. For this reason, relatively large-sized kernels such as 17×5 are used on 36-bins/octave CQT in [57]. A special case is to use different shapes of kernels in the same layer as in the Inception module [73] which is used for hit song prediction in [74].

Another question would be then how big can a kernel be? One should note that a kernel does not allow an invariance within it. Therefore, if a large target pattern may slightly vary inside, it would better be captured with stacked convolutional layers with subsampling so that small distortions can be allowed. More discussions on the kernel shapes for MIR research are available in [75, 76].

Max-pooling is frequently used in MIR to add time/frequency invariant. Such a subsampling is necessary in the DNNs for time-invariant problems in order to yield a one, single prediction for the whole input. In this case, One can begin with specifying the size of target output, followed by deciding subsampling details (how many and how much in each stage) somehow empirically.

A special use-case of the convolutional layer is to use 1D convolutional layers directly onto an audio signal (which is often referred as a raw input) to learn the time-frequency conversions in [4], and furthermore, [77]. This approach is also proposed in speech/audio and resulted in similar kernels learned of which fundamental frequencies are similar to log- or mel-scale frequencies [78]. Figure 2.9 illustrates a subset of trained kernel in [4]. Note that unlike STFT kernels, they are not pure sinusoid and include harmonic components.

The convolutional layer has been very popular in MIR. A pioneering convnet research

for MIR is convolutional deep belief networks for genre classification [79]. Early works relied on MFCC input to reduce computation [80], [81] for genre classification. Many works have been then introduced based on time-frequency representations e.g., CQT for chord recognition [57], guitar chord recognition [82], genre classification [83], transcription [58], mel-spectrogram for boundary detection [84], onset detection [55], hit song prediction [74], similarity learning [85], instrument recognition [86], music tagging [4], [15], [8], [17], and STFT for boundary detection [32], vocal separation [87], and vocal detection [88]. One-dimensional CNN for raw audio input is used for music tagging [4], [77], synthesising singing voice [89], polyphonic music [90], and instruments [91].

One of the limitations of these convolution neural network-based approaches is lack of scaling property. Convolutional operation does not automatically provide the shift invariance, which is, to some extent, desired along the time axis of time-frequency representations of music content.

2.3.6 Recurrent layers

A recurrent layer incorporates a recurrent connection and is formulated as Eq. 2.7.

$$\begin{aligned} y_t &= f_{out}(\mathbf{V}h_t), \\ h_t &= f_h(\mathbf{U}x_t + \mathbf{W}h_{t-1}), \end{aligned} \tag{2.7}$$

where f_h is usually tanh or ReLU, f_{out} can be softmax/sigmoid/etc., h_t: hidden vector of the network that stores the information at time t, and $\mathbf{U}, \mathbf{V}, \mathbf{W}$ are matrices which are trainable weights of the recurrent layer. To distinguish the RNN with this formula from other variants of RNNs, it is often specified as vanilla RNN.

An easy way to understand recurrent layers is to build them up from dense layers. If the networks in Figure 2.11 (b) are isolated by each time step (i.e., if $\mathbf{W}$ is discon-

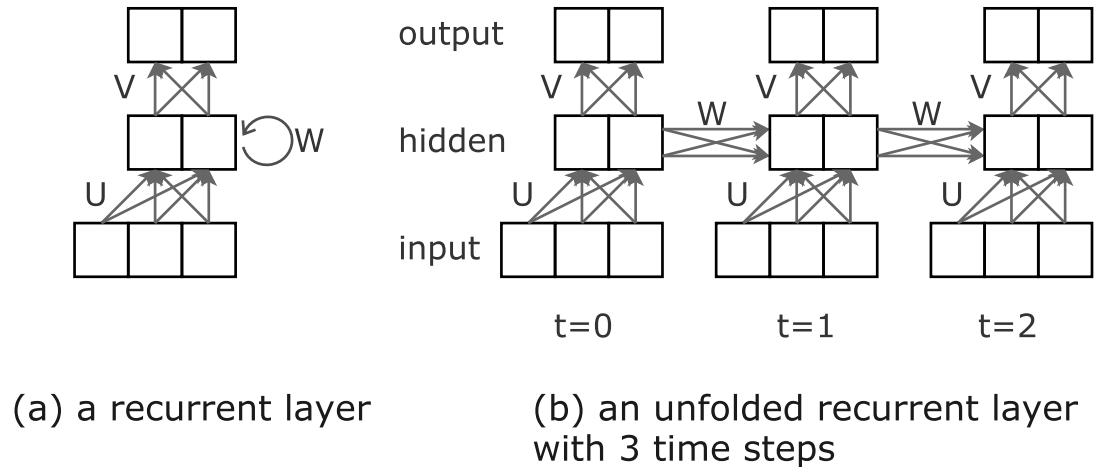

Figure 2.11: Illustrations of a recurrent layer as (a) folded and (b) unfolded, with the dimensionality of input/hidden/output as 3/2/2. Note that $\mathbf{W} \in \mathbb{R}^{2\times 2}$ and fully-connects the hidden nodes between time-steps.

nected), the network becomes a feed-forward network with two dense layers which are parametrised with two matrices, $\mathbf{V}$ and $\mathbf{U}$. Let's further assume that there are three data samples, each of which is a pair ($x \in \mathbb{R}^3$, $y \in \mathbb{R}^2$) and is fed to the isolated network one by one. In this case, only the relationship between x and y for data sample is modelled by V and U and one is assuming that there is no relationship between data samples, e.g., input at $t = 0$ is not related to the outputs at $t \neq 0$. By connecting them with the recurrent connection $\mathbf{W}$, the relationships between x at $t = 0$ and y's at $t = 0, 1, 2$ are modelled by U, V, W. In other words, the network learns how to update the 'memory' from the previous hidden state (h_{t-1}) to the current hidden state (h_t) which is then used to make a prediction (y_t).

In short, recurrent layer models $p(y_t|x_{t-k}, ..., x_t)$. This is similar to the goal of HMMs (hidden Markov models). Compared to HMMs which consist of 1-of-n states, RNNs are based on hidden states that consist of continuous value and therefore scale with a large amount of information.

In practice, modified recurrent units with gates have been widely used. A **gate** is a vector-multiplication node where the input is multiplied by a same-sized vector to

attenuate the amount of input. Gated recurrent networks usually use long short-term memory units (LSTM, [36]) and gated recurrent unit (GRU, [92]). In LSTM units, the gates control how much read/write/forget from the memory h. Moreover, additive connections between time-steps help gradient flow, remedying the vanishing gradient problem [45]. RNNs with gated recurrent units have been achieving state-of-the-art results in many sequence modelling problems such as machine translation [93] and speech recognition [94] as well as MIR problems, e.g., singing voice detection [95].

Sometimes, an RNN has another set of hidden nodes that are connected by W but with a reversed direction. This is called a bi-directional RNN [96] and can model the recurrence both from the past and the future, i.e., $p(y_t|x_{t-k}, ..., x_t)$ and $p(y_t|x_{t+1}, ..., x_{t+k})$. As a result, the output is a function of its past and future inputs. An intuitive way of understanding it is to imagine another recurrent layer in parallel that works in the reversed time order.

2.3.6.1 Recurrent layers and music

Since the input is fully-connected to the hidden layer with U in a recurrent layer, a recurrent layer can replace with a dense layer with contextual inputs. Figure 2.8 - **r1** is a many-to-many recurrent layer applied to a spectrogram while **r2** is a many-to-one recurrent layer which can be used at the final layer.

Since the shift invariance cannot be incorporated in the computation inside recurrent layers, recurrent layers may be suitable for the sequences of features. The features can be either known music and audio features such as MFCCs or feature maps from convolutional layers [8].

All the inputs are sequentially transformed and summed to a V-dimensional vector, therefore it should be capable of containing enough information. The size, V, is one of the hyperparameters and choosing it involves many trials and comparison. Its initial value can be estimated by considering the dimensionality of input and output. For example,

V can be between their sizes, assuming the network learns to compress the input while preserving the information to model the output.

One may want to control the length of a recurrent layer to optimise the computational cost. For example, on the onset detection problem, probably only a few context frames can be used since onsets can be specified in a very short time, while chord recognition may benefit from longer inputs.

Many time-varying MIR problems are time-aligned, i.e., the ground truth exists for a regular rate and the problem does not require sequence matching techniques such as dynamic time warping. This formulation makes it suitable to simply apply 'many-to-many' recurrent layer (Figure 2.8 - **`r1`**). On the other hand, classification problems such as genre or tag only have one output prediction. For those problems, the 'many-to-one' recurrent layer (Figure 2.8 - **`r2`**) can be used to yield only one output prediction at the final time step.

For many MIR problems, inputs from the future can help the prediction and therefore bi-directional setting is worth trying. For example, onsets can be effectively captured with audio contents before and after the onsets, and so are offsets/segment boundaries/beats.

So far, recurrent layers have been mainly used for time-varying prediction; for example, singing voice detection [95], singing and instrument transcription [69], [58] [97], and emotion prediction [98]. For time-invariant tasks, a music tagging algorithm used a hybrid structure of convolutional and recurrent layers [8].

2.4 Compact-convnet

Along with this thesis, a simple 5-layer 2D convnet structure with 3×3 kernels and 32-channels per layer, which is similar to the network in [15] and called Compact-convnet, is used. The structure is illustrated in Figure 2.12. It assumes an input size of 96×1360,

Table 2-B: Details of the Compact-convnet architecture. 2-dimensional convolutional layer is specified by (channel, (kernel lengths in frequency, time)). Pooling layer is specified by (pooling length in frequency and time). Batch normalization layer [99] and exponential linear unit activation [100] are used after all convolutional layers.

input (1, 96, 1360)	
Conv2d and Max-Pooling	(32, (3, 3)) and (2, 4)
Batch normalization - ELU activation	
Conv2d and Max-Pooling	(32, (3, 3)) and (4, 4)
Batch normalization - ELU activation	
Conv2d and Max-Pooling	(32, (3, 3)) and (4, 5)
Batch normalization - ELU activation	
Conv2d and Max-Pooling	(32, (3, 3)) and (2, 4)
Batch normalization - ELU activation	
Conv2d and Max-Pooling	(32, (3, 3)) and (4, 4)
Batch normalization - ELU activation	
Fully-connected layer	(50)
output (50)	

which is reduced to 1×1 after 5 convolutional layers. The 32-dimensional feature map is mapped to 50-dimensional output layer, where each node represents the probability of each label. More details are presented in Table 2-B with its activation functions and batch normalization layers. In Chapter 4, this structure is called *k2c2* to discriminate itself from similar but different convnet structures in terms of kernel and convolution dimensionalities.

Later in this thesis (Chapter 4), I proposed CRNN, convolutional recurrent neural networks, which show similar to or better performance than usual convnet structures such as Compact-convnet. I still preferred to use Compact-convnet for several reasons: i) CRNN is widely considered to be a variant of convnet, especially the recurrent layer only plays a role of effectively aggregating the feature maps out of convolutional layers and ii) pure convnets are more stable to train while showing equal performance per training time.

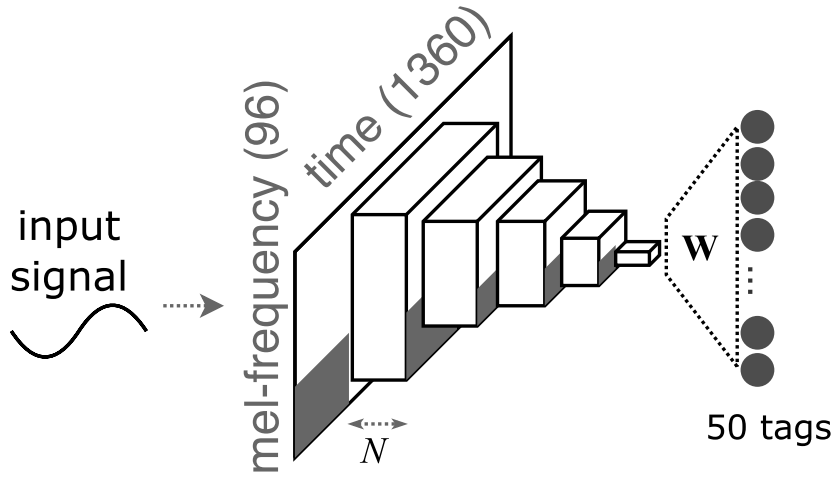

Figure 2.12: Compact-convnet structure

2.5 Summary

In this chapter, I presented backgrounds of machine learning and deep learning. They are explained in the context of the application for MIR problems with visualisations to understand their mechanism, which will help to understand many design choices of the networks used in this thesis and in general. At the end of the chapter, the 'compact convnet' was introduced, which is used for multiple times through this thesis.

Chapter 3

The Dataset, Label, and Noise

Abstract

In recent research that uses machine learning methods, it is important to understand the training dataset to expect and understand the behaviour of the trained classifiers or deep learning models. This chapter introduces my work on investigating the music tagging dataset in order to build a deeper understanding of my other works on music tagging.

In this chapter, I present a quantification of the label noise in the groundtruth of the Million Song Dataset as well as measuring its effect on the training and evaluation. During the quantification, I introduce a concept named 'tagability' to explain my observation on the noise and its effect.

This chapter is closely related to my IEEE Journal paper, "The Effects of Noisy Labels on the Deep Convolutional Neural Networks for Music Classification" [7].

3.1 Introduction

Under a supervised learning scheme, many music classification works including my own ones ([15] and [8]) assume that the training data provides correct information – the pairwise information of music items and its labels. In classification, the label is given as binary, *1 (=True)* or *0 (=False)*. However, as will be elaborated in this chapter, in music tagging datasets, *False* can mean either *Negative* or *Unknown*. This is due to the *weak labelling*, which introduces a particular type of label noise to the negative labels. Given the dataset be noisy, a research question would be "how bad the noise is?" which will be followed by "what is its impact on the system?". These are the questions I pursue to answer in this chapter.

3.1.1 Labelling strategy

Audio items in a dataset may be 'strongly' or 'weakly' labelled, which may refer to several different aspects of tagging. A particular form of weak labelling means that only positive associations between tags and tracks are provided. This means, given a finite set of tags, a listener (or annotator) applies a tag in case s/he recognises a relationship between the tag and the music. In this scenario, no negative relations are provided, and as a result, a tag being positive means it is 'true', but a tag being negative, i.e. not applied, means 'unknown'. Typical crowd-sourced datasets are weakly labelled, because it is the only practical solution to create a large dataset. Furthermore, listeners in online platforms cannot be reasonably expected to provide negative labels given the large number of possible tags. Strong labelling in this particular context would mean that disassociation between a tag and a track confirms negation, i.e., a zero element in a tag-track matrix would signify that the tag does not apply. To the best of my knowledge, *CAL500* [101] is the biggest music tag dataset (500 songs) that is strongly labelled. Most recent research has relied on collaboratively created, and therefore weakly-labelled datasets such as *MagnaTagATune* [102] (5,405 songs) and the *Million Song Dataset (MSD)* [103] containing 505,216 songs if only tagged items are counted.

Learning from noisy labels is an important problem, therefore several studies address this in the context of conventional classifiers such as support vector machines [104]. In deep learning research, [105] assumes a binary classification problem while [106] deals with multi-class classification. Usually, auxiliary layers are added to learn to fix the incorrect labels, which often requires a noise model and/or an additional clean dataset. Both solutions, together with much other research, are designed for single-class classifications, and there is no existing method that can be applied for music tagging when considered as multi-label classification and when the noise is highly skewed to negative labels. This will be discussed in Section 3.2.2.

3.1.2 Tagging dataset and label noise

In this chapter, I perform data-driven analyses, focussing on the effects of noisy labels on deep convolutional neural networks for automatic tagging of popular music. Label noise is unavoidable in most real-world applications, therefore it is crucial to understand its effects. I hypothesise that despite the noise, neural networks are able to learn meaningful representations that help to associate audio content with tags and show that these representations are useful even if they remain imperfect. The insights provided by my analyses may be relevant and valuable across several domains where social tags or folksonomies are used to create automatic tagging systems or in research aiming to understand social tags and tagging behaviour. The primary contributions of this chapter are as follows: *i)* An analysis of the largest and most commonly used public dataset for music tagging including an assessment of the distribution of labels within this dataset. *ii)* I validate the ground truth and discuss the effects of noise, e.g. mislabelling, on both training and evaluation.

Finally, I draw overall conclusions and discuss cross-domain applications of my methodology in Section 3.3.

3.1.3 The dataset

I select the Million Song Dataset (MSD) [103] as it is the largest public dataset available for training music taggers. Other large datasets are MagnaTagATune [102] and CAL10K [107] which have 25,863 and 10,271 tracks respectively. MSD is weakly labelled and therefore potentially has the problem that I am concerning in this chapter. MSD also provides crawlable track identifiers for audio signals, which enables me to access the audio and re-validate the tags manually by listening. MSD is a foundational dataset through my PhD research and many experiments in this thesis are based on it.

The tags in the MSD are collected using the Last.fm API which provides access to

crowd-sourced music tags. Because the listeners are allowed to use any text as tags, the original tag data is extremely diverse – there are 505,216 tracks with at least one tag and 522,366 unique tags in the dataset.

In this thesis, I use the top 50 tags sorted by popularity (occurrence counts) in the dataset. The number of tags, '50', may sound a tentative choice, but it is chosen to compromise between two contradictory goals - to maximise the number of tags to utilise richer information during training and to minimise the number of tag due to the long-tail of tag distribution. The tags include 28 genres ('rock', 'pop', 'jazz', 'funk'), 5 eras ('60s' – '00s'), 5 instruments ('guitar', 'female vocalists'), and 12 moods ('sad', 'happy', 'chill'). There are 242,842 clips with at least one of the top 50 tags. The tag counts range from 52,944 ('rock') to 1,257 ('happy') and there are 12,348 unique tag vectors represented as a joint variable of 50 binary values.

3.2 Experiments and Discussions

3.2.1 Tag co-occurrences in the MSD

In this section, the distribution and mutual relationships of tags in the dataset are investigated. This procedure helps to understand the task. Furthermore, the analysis represents information embedded in the training data. This will be compared to knowledge we can extract from the trained network (see Chapter 7).

Here, I investigate the tuple-wise[1] relations between tags and plot the resulting normalised co-occurrence matrix (NCO) denoted $\mathbf{C}$. Let me define $\#y_i \coloneqq |\{(x, y) \in D | y = y_i\}|$, the total number of the data points with i^{th} label being *True* given the dataset D where (x, y) is an (input, target) pair. In the same manner, $\#(y_i \wedge y_j)$ is defined as the number of data points with both i^{th} and j^{th} labels being *True*, i.e., those two tags

[1] These are not pairwise relations since there is no commutativity due to the normalisation term.

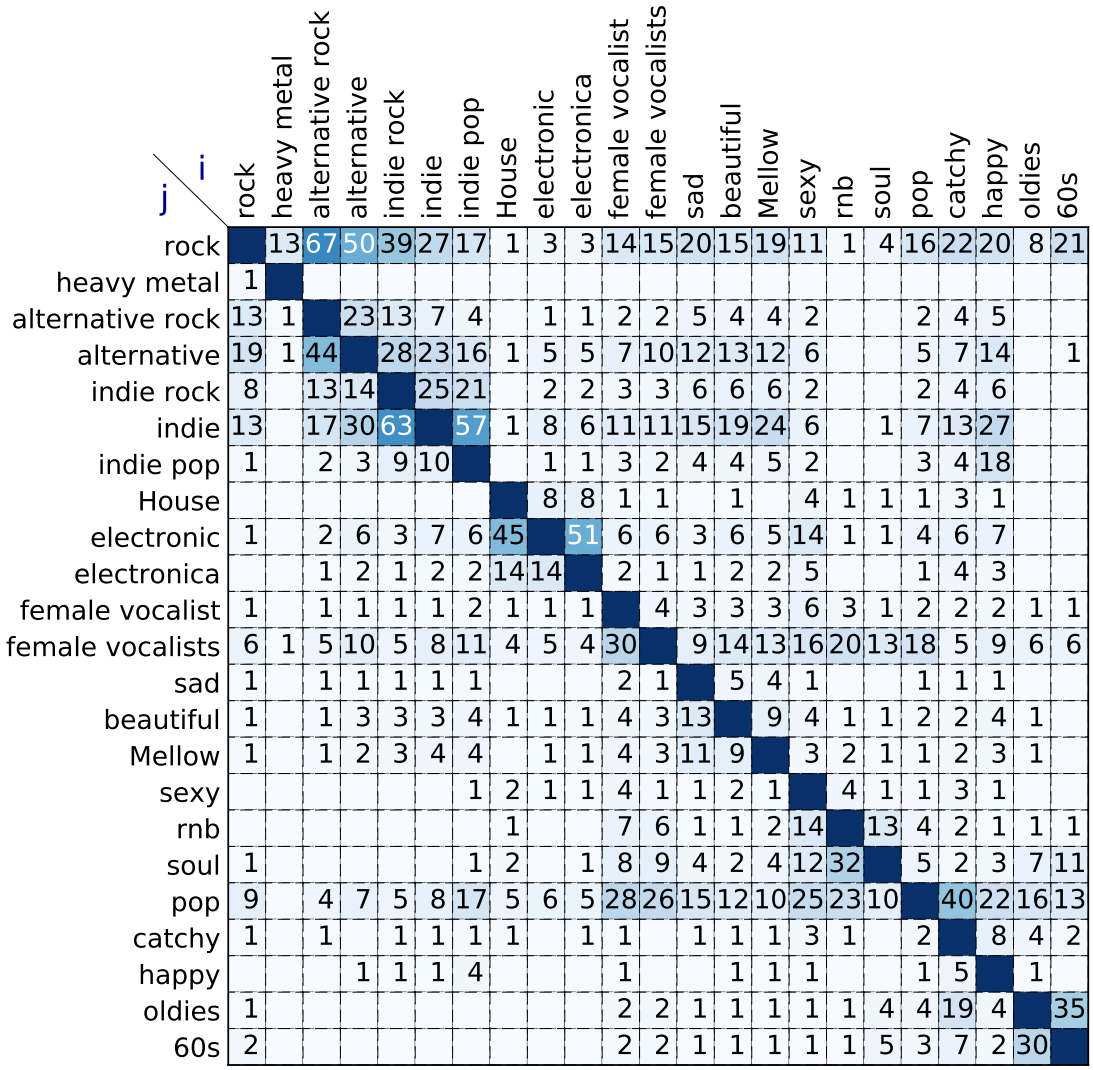

Figure 3.1: Normalised tag co-occurrence pattern of the selected 23 tags from the training data. For the sake of visualisation, I selected 23 tags out of 50 that have high co-occurrences and represent different categories; genres, instruments and moods. The values are computed using Eq. 3.1 (and are multiplied by 100, i.e., shown in percentage), where y_i and y_j respectively indicate the labels on the x-axis and y-axis.

co-occur. NCO is computed using Eq. 3.1 and illustrated in Fig. 3.1.

$$C(i,j) = \#(y_i \wedge y_j)/\#y_i. \tag{3.1}$$

In Fig.3.1, the x- and y-axes correspond to i, j respectively. Note that $C(i, j)$ is not symmetric, e.g., ('alternative rock', 'rock') = #(alternative rock∧rock)/#alternative rock.

The pattern in NCO reveals mutual tag relationships which I categorise into three types: *i)* tags belonging to a genre hierarchy, *ii)* synonyms, i.e., semantically similar words and *iii)* musical similarity. Genre hierarchy tuples include for instance ('alternative rock', 'rock'), ('house', 'electronic'), and ('heavy metal', 'metal'). All the first labels are sub-genres of the second. Naturally, we can observe that similar tags such as ('electronica', 'electronic') are highly likely to co-occur. Lastly, I notice tuples with similar meaning from a musical perspective including ('catchy', 'pop'), ('60s', 'oldies'), and ('rnb', 'soul'). Interestingly, $C(i, j)$ values with highly similar tag pairs, including certain subgenre-genre pairs, y_i and y_j are not close to 100% as one might expect. For example the pairs ('female vocalist', 'female vocalists')and ('alternative', 'alternative rock') reach only 30% and 44% co-occurrence values, while the pairs ('rock' and 'alternative rock'), ('rock', 'indie rock') reach only 69% and 39% respectively. This is primarily because *i)* items are weakly labelled and *ii)* there is often a more preferred tag to describe a certain aspect of a track compared to others. For instance, 'female vocalists' appears to be preferred over 'female vocalist' in the data as also noted in [6]. The analysis also reveals that certain types of label noise related to missing tags or taxonomical heterogeneity turn out to be very high in some cases. For instance, only 39% of 'indie rock' tracks are also tagged 'rock'. The effect of such label noise is studied more deeply in Section 3.2.2. Furthermore, the computed NCO under-represents these co-occurring patterns. This effect is discussed in the the following section (section 3.2.2) and Chapter 7.

3.2.2 Validation of the MSD as ground truth for auto-tagging

Next, I move to the main body of this chapter, analysing the ground truth noise in the MSD and examine its effect on training and evaluation. There are many sources of noise including incorrect annotations as well as information loss due to the trimming of full tracks to preview clips. Some of these factors may be assumed to be less adverse

Table 3-A: The scores of ground truth with respect to the strongly-labelled manual annotation (subset100) in (a)-(d) and occurrence counts by the ground truth (e), estimation (f), and on Subset400.

	Scores of the ground truth on Subset100				Occurrence counts		
	(a) Error rate, Positive label [%]	(b) Error rate, Negative label [%]	(c) Precision [%]	(d) Recall [%]	(e) In ground truth (for all items)	(f) Estimate by Eq.3.2 and Subset100	(g) By our annotation (on Subset400)
instrumental	6.0	12.0	94.0	88.7	8,424 (3.5%)	36,048 (14.9%)	85 (21.3%)
female vocalists	4.0	24.0	96.0	80.0	17,840 (7.3%)	71,127 (29.3%)	94 (23.5%)
male vocalists	2.0	64.0	98.0	60.5	3,026 (1.2%)	156,448 (64.4%)	252 (64.0%)
guitar	2.0	70.0	98.0	58.3	3,311 (1.4%)	170,916 (70.4%)	266 (66.5%)

than others. In large-scale tag datasets, the frequently used weak labelling strategy may introduce a significant amount of noise. This is because by the definition of weak labelling, considering numerous tags a large portion of items remain unlabelled, but then these relations are assumed to be negative during training as I have discussed earlier in this chapter.

Validation of the annotation requires re-annotating the tags after listening to the excerpts, which is not a trivial task for several reasons. First, manual annotation does not scale and requires significant time and effort. Second, there is no single correct answer for many labels – music genre is an ambiguous and idiosyncratic concept, emotion annotation is highly subjective too, so as labels such as 'beautiful' or 'catchy'. Instrumentation labels can be objective to some extent, assuming the annotators have expertise in music. Therefore, I re-annotate items in two subsets using four instrument labels as described below.

- **Labels:** 'instrumental', 'female vocalists', 'male vocalists', 'guitar'.
- **Subsets:**
  - Subset100: randomly selected 100 items for each class. All are from the training set and positive/negative labels are balanced as 50/50 respectively.
  - Subset400: randomly selected 400 items from the test set.

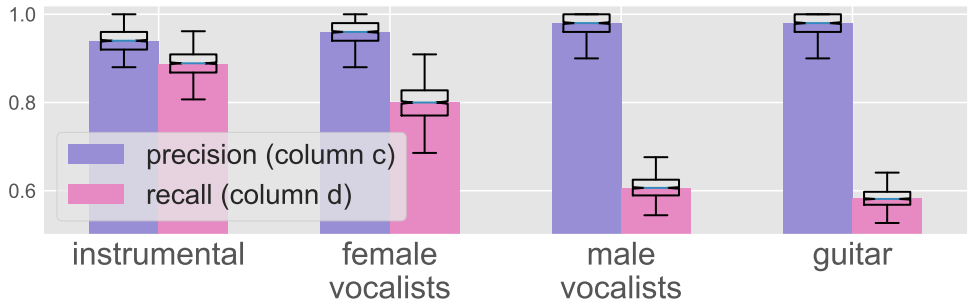

Figure 3.2: The precision and recall of ground truth on Subset100, corresponding to columns (c) and (d) in Table 3-A. They are plotted with 95% confidential interval computed by bootstrapping [108].

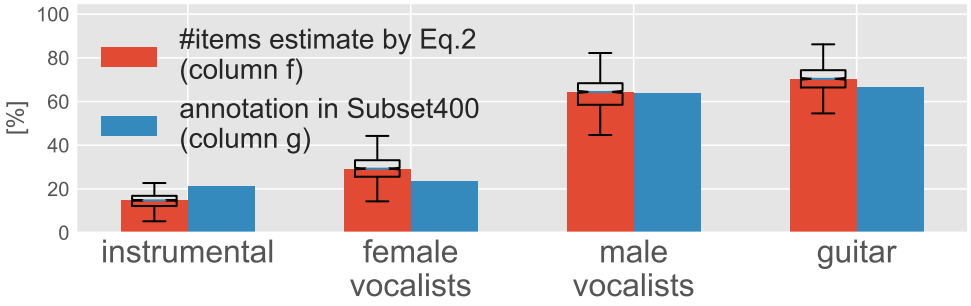

Figure 3.3: The estimates of the number of items (red) and the number of items in Subset400 (blue), both in percentage (they correspond to columns (f) and (g) in Table 3-A). The estimates are plotted with 95% confidential interval computed by bootstrapping [108].

3.2.2.1 Measuring label noise and defining tagability

Table 3-A column (a)-(d) summarises the statistics of Subset100. Confidence intervals for precision and recall are computed by bootstrapping [108] and plotted in Figure 3.2. The average error rate of negative labels is 42.5%, which is very high, while that of positive labels is 3.5%. Equivalently, the precision of the ground truth is high (96.5% on average) while the recall is much lower (71.9% on average). This suggests that the tagging problem should be considered weakly supervised learning to some extent. We may expect this problem exists in other weakly-labelled datasets as well.

Such a high error rate for negative labels suggests that the tag occurrence counts

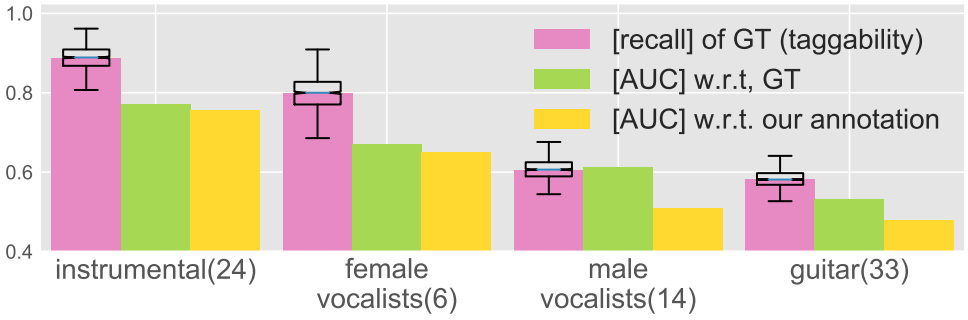

Figure 3.4: The recall rates (tagability, pink), AUC scores with respect to the GT (ground truth) (green), and AUC scores with respect to my annotation (yellow), all reported on Subset400. The numbers on the x-axis labels are the corresponding popularity rankings of tags out of 50. The recall rates and their 95% confidential intervals are identical to Figure 3.2 but plotted again for comparison with tag-wise AUC scores.

in the ground truth are significantly under-represented. This can be related to the *tagability* of labels, a notion which may be defined as *the likelihood that a track will be tagged positive for a label when it really is positive*. If the likelihood is replaced with the proportion of items, tagability is measured by recall as presented in Table 3-A as well as in Figure 3.2. For example, bass guitar is one of the most widely used instruments in modern popular music, but it is only the 238^{th} most popular tag in the MSD since tagging music with 'bass guitar' does not provide much information from the perspective of the average listener. Given the scores, we may assume that 'female vocalists' (88.7% of recall) and 'instrumental' (80.0%) are more 'tagable' than 'male vocalists' (60.5%) and 'guitar' (58.3%), which indicates that the latter are presumably considered less unusual.

The correct number of *positive / negative* items can be estimated by applying Bayes' rule with the error rate. The estimated positive label count $\hat{N}^+$ is calculated using Eq.3.2 as follows:

$$\hat{N}^+ = N^+(1 - p^+) + (T - N^+)p^-, \tag{3.2}$$

where N^+ is the tag occurrence, T is the number of total items ($T = 242,842$ in this case), and p^+, p^- refers to the error rates of positive and negative labels respectively.

Column (f) of Table 3-A and Figure 3.3 present the estimated occurrence counts using Equation 3.2. This estimate is validated using Subset400. Comparing the percentages in columns (f) and (g) confirms that the estimated counts are more correct than the tag occurrences of the ground truth. For all four tags, the confidence intervals overlap with the percentage counted in Subset400 as illustrated in Figure 3.3. In short, the correct occurrence count is not correlated with the occurrence in the dataset, which shows the bias introduced by tagability. For instance, 'male vocalists' is more likely to occur in music than 'female vocalists', which means it has lower tagability, and therefore it ends up having fewer occurrences in the ground truth.

3.2.2.2 Effects of incorrect ground truth on the training

Despite such inaccuracies, it is possible to train networks for tagging with good performances using the MSD, achieving an AUC between 0.85 [15] and 0.90 [17]. This may be because even with such noise, the network is weakly supervised by stochastically correct feedbacks, where the noise is alleviated by a large number of training examples [109]. In other words, it is believed in general that given $\mathbf{x}$ is the input and $\mathbf{y}_{true}$, $\mathbf{y}_{noisy}$ are the correct and noisy labels respectively, the network can approximate the relationship $f : \mathbf{x} \rightarrow \mathbf{y}_{true}$ when training using $(\mathbf{x}, \mathbf{y}_{noisy})$.

However, I suggest that the noise affects the training and it is reflected in the performances of different tags. In [8], where different deep neural network structures for music tagging were compared, I observed a pattern of the per-tag performances that is common among different structures. This is illustrated in Figure 3.5 where the x-axis labels represent tag popularity ranking. The performances were not correlated with the ranking, the reported correlation is 0.077, therefore the question remained unanswered in [8].

I conjecture that tagability, which is related to (negative) label noise can explain tag-wise performance differences. It is obvious that a low tagability implies more false

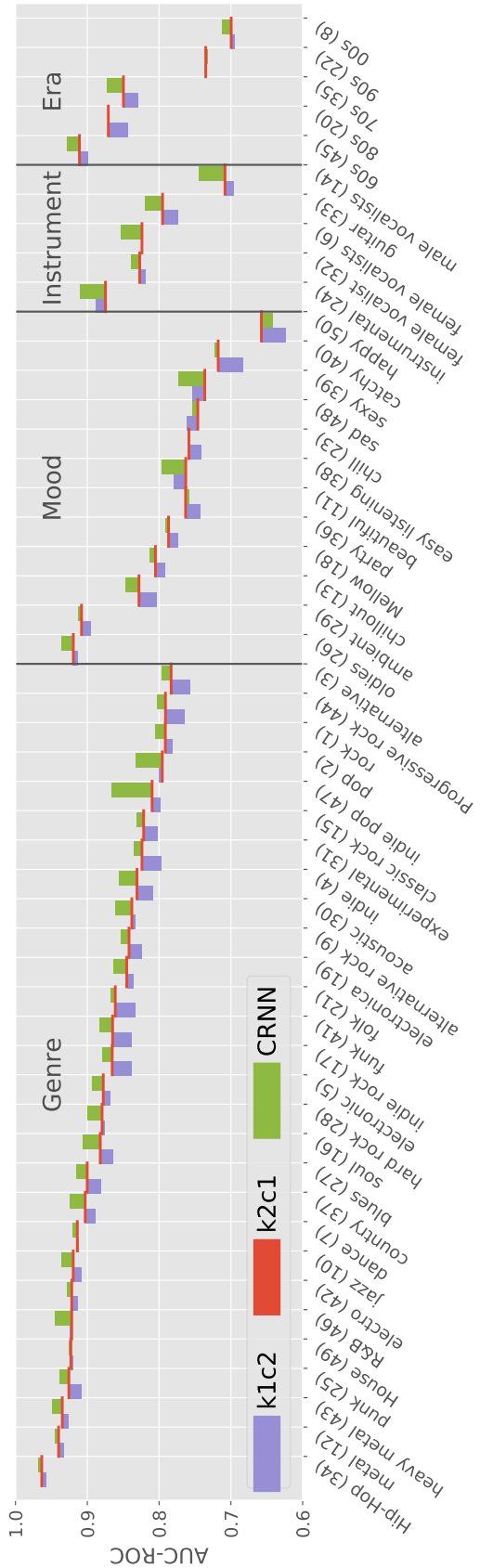

Figure 3.5: k1c2, k2c1, and CRNN are the names of network structures in [8]. *i*) AUC of each tag is plotted using a bar chart and line. For each tag, the red line indicates the score of k2c1 which is used as a baseline of bar charts for k1c2 (blue) and CRNN (green). In other words, the blue and green bar heights represent the performance gaps, k2c1-k1c2 and CRNN-k2c1, respectively. *ii*) Tags are grouped by categories (genre/mood/instrument/era) and sorted by the score of k2c1. *iii*) The number in parentheses after each tag indicates that tag's popularity ranking in the dataset.

negatives in the ground truth, and as a result, we feed the network with more confusing training examples. For example, assuming there is a pattern related to male vocalists, the positive-labelled tracks provide mostly correct examples. However, many examples of negative-labelled tracks (64% in Subset100) also exhibit the pattern. Consequently, the network is forced to distinguish hypothetical differences between the positive-labelled true patterns and the negative-labelled true patterns, which leads to learning a more fragmented mapping of the input. This is particularly true in music tagging datasets where negative label noise dominates the total noise. This is supported by data both in this chapter and [8] as discussed below.

First, tagabilities (or recall) and AUC scores with respect to the ground truth and my re-annotation are plotted in Figure 3.4 using Subset400 items and Compact-convnet 2.4. Both AUC scores are positively correlated to tagability while they are not related to the tag popularity rankings. Although the confidence intervals of 'instrument' vs. 'female vocalists', and 'male vocalists' vs. 'guitar' overlap, there is an obvious correlation. The performances on the whole test set also largely agree with my conjecture.

Second, in Figure 3.5, AUC scores for instrument tags are ranked as 'instrumental' > 'female vocalists' > 'guitar' > 'male vocalists' for all three convnet structures. This aligns with tagability in Figure 3.4 within the confidence intervals.

This observation motivates me to expand this approach and assess tags in other categories. Within the Era category in Figure 3.5, performance is negatively correlated with the popularity ranking (Spearman correlation coefficient$=-0.7$). There is a large performance gap between old music groups (60s, 80s, 70s) and the others (90s, 00s). I argue that this may also be due to tagability. In the MSD, older songs (e.g. those released before the 90s) are less frequent compared to modern songs (90s or 00s). According to the year prediction subset of the MSD[2], 84% of tracks are released after 1990. This is also related to the fact that the tag 'oldies' exists while its opposite does not. Hence, old eras seem more tagable, which might explain the performance differences in Era tags. I

[2] `https://labrosa.ee.columbia.edu/millionsong/pages/tasks-demos`

cannot extend this approach to mood and genre tags because the numbers of tags are much larger and there may be aspects contributing to tag-wise performance differences other than tagability. The high subjectivity of many tags are also one of the aspects to make it difficult, which is also discussed in the following section.

3.2.2.3 Validation of the evaluation

Another problem with using a noisy dataset is the evaluation. In the previous section, I assumed that the system can learn a *denoised* relationship between music pieces and tags, $f : \mathbf{x} \rightarrow \mathbf{y}_{true}$. However, the evaluation of a network with respect to $\mathbf{y}_{noisy}$ includes errors due to noisy ground truth. This raises the question of the reliability of the results. I use the strongly-labelled annotation of Subset400 to assess this reliability.

Let us re-examine Figure 3.4. All AUC scores with respect to my annotation are lower than the scores with respect to the ground truth. Performance for the *guitar* tag is remarkably below 0.5, the baseline score of a random classifier. However, the overall trend of tag-wise performance does not change. Because the results are based only on four classes and a subset of songs, a question arises: How does this result generalise to other tags?

To answer the question, three AUC scores are plotted in Figure 3.6: *i)* the scores of the four instrument tags with respect to my annotation (dotted red), *ii)* the scores of the four instrument tags with respect to the given ground truth (dashed blue), and *iii)* the scores of all tags with respect to the given ground truth (solid yellow).

Before discussing the details, the limitations of this approach should be clarified. Without an ideal, complete and correct ground truth (which will make this chapter obscure), re-evaluating the evaluation cannot be noise-free. Furthermore, because of the subjectivity of majorities of the tags, it is not even possible to strongly-label the groundtruth of a subset of items. Under this circumstance, the re-evaluation is performed indirectly by second-order: using correlation, I will measure two similarities, assuming

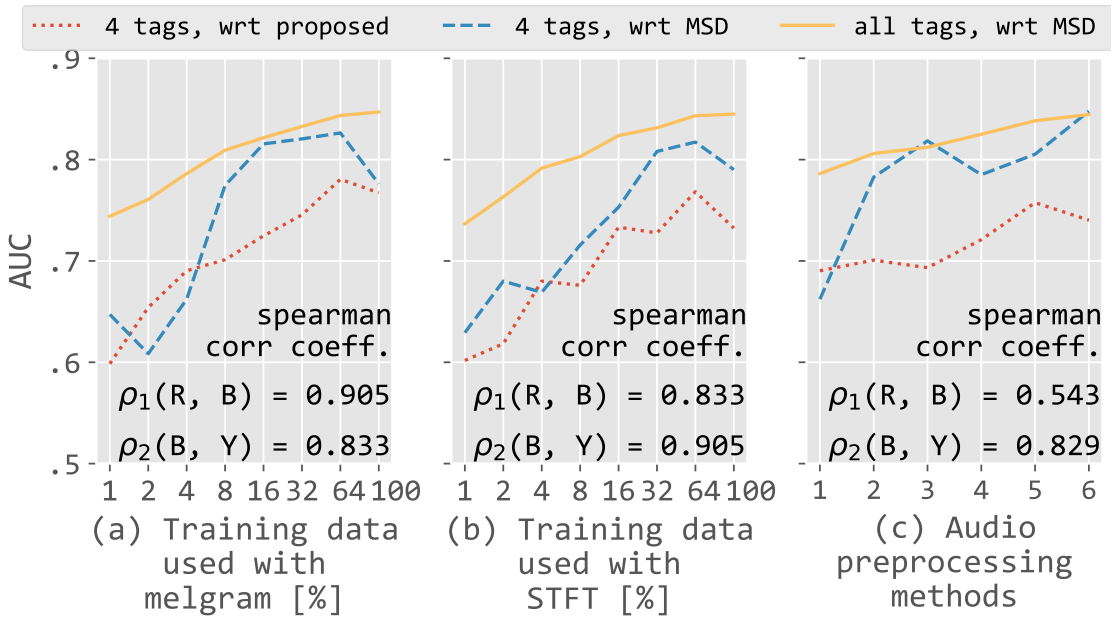

Figure 3.6: AUC scores of all tags (yellow, solid) and four instrumentation tags. Instrumentation tags are evaluated using *i)* dataset ground truth (blue, dashed) and *ii)* my strong-labelling re-annotation (red, dotted). Pearson correlation coefficients between {red vs. blue} and {blue vs. yellow} is annotated on each chart as ρ. In (c), x-axis is experiment index and various audio preprocessing methods were applied for each experiment.

both of them being high means the our re-evaluation on the subset (with limited songs and limited tags) may apply for the whole set (all songs and tags).

The reliability of evaluation is typically assessed with a given ground truth and can be measured by ρ_1, the Pearson correlation coefficient between AUCs using my annotation and the MSD. The correlation between the four tags and all other tags (shown in blue and yellow), denoted ρ_2, is a measure of how we can generalise my re-annotation result to those concerning all tags.

I selected three sets of tagging results to examine and plotted these in Figure 3.6 (a-c). The first two sets shown in subfigure (a) and (b) are results after training the Compact-convnet with varying training data size and different audio representations: (a) mel-spectrogram and (b) short-time Fourier transform. The third set of curves in Figure 3.6

(c) compare six results with varying input time-frequency representations, training data size and input preprocessing techniques including normalisation and spectral whitening. The third set is selected to observe the correlations when the performance differences among systems are more subtle than those in (a) and (b).

First, the ρ_1 values in (a)–(c) suggest that noisy ground truth provides reasonably stable evaluation for the four instrument tags. On the first two sets in (a) and (b), the scores of four tags using the MSD ground truth (in blue) are highly correlated ($\rho_1 = 0.905$ and 0.833) to the scores using my annotation (red). This suggests the evaluation using noisy labels is still reliable. However, in (c), where the scores of all tags with given ground truth (yellow) are in a smaller range, the correlation between all tags and the four tags (ρ_1) decreases to 0.543. The results imply that the distortion on the evaluation using the noisy ground truth may disguise the performance difference between systems when the difference is small. Second, large ρ_2 indicates that my validation is not limited to the four instrument tags but may be generalised to all tags. The correlation coefficients ρ_2 is stable and reasonably high in (a)–(c). It is 0.856 on average.

It is noteworthy that there are small but clear differences between models themselves in (c) unlike (a) or (b), where an identical model is used with only changing the amount of training data. In (c), the models consist of same convolutional layers but different input data type, possibly making the models learn different characteristics. This means there are not only the noise in the evaluation but also the noise introduced by the system differences, making the experiment (c) less robust than (a) and (b).

3.3 Summary

In this Chapter, I investigated several aspects how noisy labels affect the training and performance of deep convolutional neural networks for music tagging. I analysed the MSD, the largest dataset available for training a music tagger from a novel perspective. I reported on a study aiming to validate the MSD as ground truth for this task. I found

that the dataset is reliable overall, despite several noise sources affecting training and evaluation.

Overall, the behaviours of the trained network were shown to be related to the property of the given labels. The analysis showed that tagability, which I measured by recall on the ground truth, is correlated to the tag-wise performance. This opened a way to explain tag-wise performance differences within other categories of tags such as era. In the analysis of the trained network, I found that the network learns more intricate relationships between tags rather than simply reproducing the co-occurrence patterns in the ground truth. The trained network is able to infer musically meaningful relationships between tags that are not present in the training data.

Although I focused on music tagging, results provide general knowledge applicable in several other domains or tasks including other music classification tasks. The analysis method presented here and the result on the tagging dataset can easily generalise to similar tasks in other domains involving folksonomies with noisy labels or tasks involving weakly labelled datasets, e.g. image tagging, object recognition in video, or environmental sound recognition, where not all sources are necessarily labelled. Future work will explore advanced methods to learn and evaluate using noisy datasets under a structured machine learning framework. Tagability can be understood from the perspective in music cognition research and should be investigated further.

So far, I have discussed if and how much we can trust the data-driven approach for music tagging. The experiment results have revealed that although there is huge noise on the label, the networks can still take advantage of the dataset. In the next chapter, I will present the training results based on the dataset with comparing many convnet structures for music tagging. One of the compared structures is one that I propose to use for the first time in music tagging, convolutional recurrent neural networks (CRNN).

Chapter 4

CRNN and Comparisons of Convolutional Neural Network Structures

Abstract

This chapter introduces one of the main body of my PhD works. There is an enormous amount of recent research papers on proposing new deep neural network structures to improve the performance. In this chapter, I also introduce a structure for music tagging. Additionally, I present a detailed set of comparative experiments of different structures for tagging. The comparison result supports my proposal while also justifying to use Compact-convnet in this thesis.

I propose to use convolutional recurrent neural networks for music tagging for the first time. CRNNs take advantage of convolutional neural networks (CNNs)[a] for local feature extraction and recurrent neural networks for temporal summarisation of the extracted features. I present not only the proposed structure's performance but also the performances of other types of deep neural networks. CRNNs show a strong performance with respect to the number of parameter and training time, indicating the effectiveness of its hybrid structure in music feature extraction and feature summarisation. [b]

The work introduced in this chapter is closely related to my ICASSP 2017 paper, "Convolutional Recurrent Neural Networks for Music Classification"" [8].

[a] I use 'CNN' instead of 'convnet' in this chapter to follow the naming convention of the proposed structure, CRNN.

[b] One of the final results is the performances per tag and per structure, which was already introduced in the previous chapter.

4.1 Introduction

CNNs have been popular in many fields including MIR. CNNs assume features that are at different levels of hierarchy and can be extracted by convolutional kernels. The hierarchical features are learned to achieve a given task during supervised training. For example, learned features from a CNN that is trained for genre classification exhibit low-level features (e.g., onset) to high-level features (e.g., percussive instrument patterns) [12]

as will be discussed in Chapter 7.

Recently, CNNs have been combined with recurrent neural networks (RNNs) which are often used to model sequential data such as audio signals or word sequences. This hybrid model is called a convolutional recurrent neural network (CRNN). A CRNN can be described as a modified CNN by replacing the last convolutional layers with a recurrent layer. In CRNNs, CNNs and RNNs play the roles of feature extractor and temporal summariser respectively. Adopting an RNN for aggregating the features enables the networks to take the global structure into account while local features are extracted by the remaining convolutional layers. This structure was first proposed in [110] for document classification and later applied to image classification [111] and music transcription [112].

CRNNs fit the music tagging task well. RNNs are more flexible in selecting how to summarise the local features than CNNs which are rather static by using an weighted sum (convolution) and subsampling. This flexibility can be helpful because some of the tags (e.g., mood tags) may be affected by the global structure while other tags such as instruments can be affected by the local and short-segment information.

In this chapter, CRNNs for music tagging are introduced and compared with three pre-existing CNNs. For correct comparisons, the hardware, data, and optimisation techniques are carefully controlled while varying two attributes of the structure: *i)* the *number of parameters* and *ii) computation time*.

4.2 Models

There are four models – `CRNN` with `k1c2`, `k2c1`, and `k2c2`, which are illustrated in Figure 4.1. The three latter models are named in a way to specify their kernel shape (e.g., `k1` for 1D kernels) and convolution dimension (e.g. `c2` for 2D convolutions). The specifications are shown in Table 4-A. For all networks, the input is assumed to be of size 96×1366

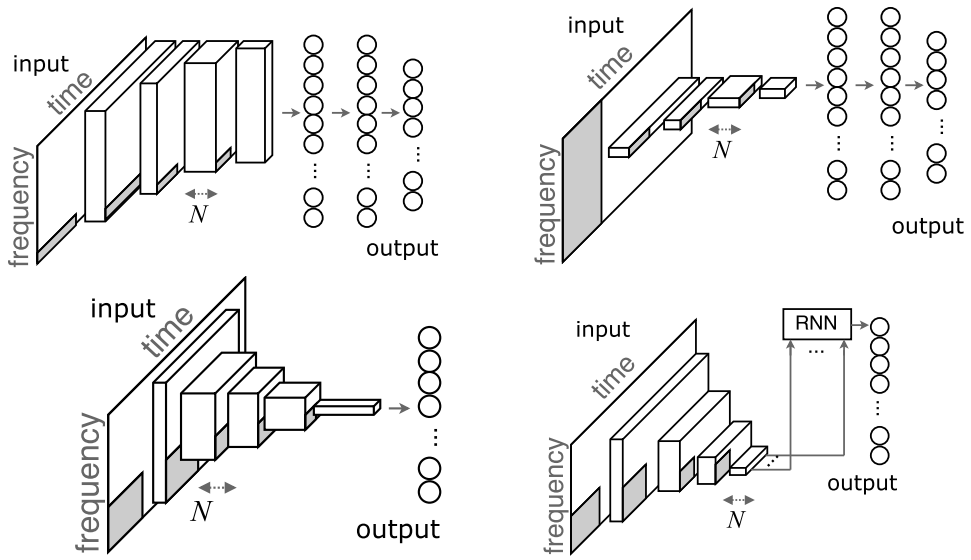

Figure 4.1: Block diagrams of `CNN (k1c2, top-left)`, `CNN (k2c1, top-right)`, `CNN (k2c2, lower-left)`, and `CRNN, lower-right`. The grey areas illustrate the convolution kernels. N refers to the number of feature maps of convolutional layers. RNN is, in the experiment, gated recurrent unit (GRU) and uni-directional.

(mel-frequency band×time frame) and single channel. Sigmoid functions are used as activation at output nodes because music tagging is a *multi*-label classification task.

Across the structures, all the convolutional and fully-connected layers are equipped with identical optimisation techniques and activation functions – batch normalization [99] and ELU activation function [100]. This is for a correct comparison since optimisation techniques greatly improve the performances of networks that are having essentially the same structure. Exceptionally, `CRNN` has weak dropout (0.1) between convolutional layers to prevent overfitting of the RNN layers [113].

4.2.1 CNN - `k1c2`

`k1c2` in Figure 4.1 is motivated by structures for genre classification [81]. The network consists of 4 convolutional layers that are followed by 2 fully-connected layers. One-dimensional convolutional layers (1×4 for all, i.e., convolution along time-axis) and max-pooling layers ((1×4)-(1×5)-(1×8)-(1×8)) alternate. Each element of the last feature

map (the output of the 4-th sub-sampling layer) encodes a feature for each band. They are flattened and fed into a fully-connected layer, which acts as the classifier.

4.2.2 CNN - k2c1

k2c1 in Figure 4.1 is motivated by structures for music tagging [4] and genre classification [83]. The network consists of 5 convolutional layers that are followed by 2 fully-connected layers. The first convolutional layer (96×4) learns 2D kernels that are applied to the whole frequency band. After then, one-dimensional convolutional layers (1×4 for all, i.e., convolution along time-axis) and max-pooling layers ((1×4) or (1×5)) alternate. The results are flattened and fed into a fully-connected layer.

This model compresses the information of whole frequency range into one band in the first convolutional layer and this helps to reduce the computation complexity vastly. At the same time, the compression is more radical than what's happening in other networks which gradually reduce the length in the frequency axis.

4.2.3 CNN - k2c2

CNN structures with 2D convolution have been used in music tagging [15] and vocal/instrumental classification [88]. k2c2 consists of five convolutional layers of 3×3 kernels and max-pooling layers ((2×4)-(2×4)-(2×4)-(3×5)-(4×4)) as illustrated in Figure 4.1. The network reduces the size of feature maps to 1×1 at the final layer, where each feature covers the whole input rather than each frequency band as in k1c1 and k2c1.

This model allows time and frequency invariances in different scale by gradual 2D sub-samplings. Also, using 2D subsampling enables the network to be *fully-convolutional*, which ultimately results in fewer parameters.

k1c2						k2c1						k2c2						CRNN					
No. params ($\times 10^6$)	0.1	0.25	0.5	1.0	3.0		0.1	0.25	0.5	1.0	3.0		0.1	0.25	0.5	1.0	3.0		0.1	0.25	0.5	1.0	3.0
Layer type	Layer width					Type	Layer width					Type	Layer width										
conv2d	15	23	33	47	81	conv1d	43	72	106	152	265	conv2d	20	33	47	67	118	conv2d	30	48	68	96	169
conv2d	15	23	33	47	81	conv1d	43	72	106	152	265	conv2d	41	66	95	135	236	conv2d	60	96	137	195	339
conv2d	30	47	66	95	163	conv1d	43	72	106	152	265	conv2d	41	66	95	135	236	conv2d	60	96	137	195	339
conv2d	30	47	66	95	163	conv1d	87	145	212	304	535	conv2d	62	100	142	203	355	conv2d	60	96	137	195	339
FC	30	47	66	95	163	conv1d	87	145	212	304	535	conv2d	83	133	190	271	473	rnn	30	48	68	96	169
FC	30	47	66	95	163	FC	87	145	212	304	535							rnn	30	48	68	96	169
						FC	87	145	212	304	535												

Table 4-A: Hyperparameters and results of all structures. Number of parameters indicates the total number of trainable parameters in the structure. Layer width indicates either the number of feature maps of a convolutional layer or number of hidden units of fully-connected/RNN layers. Max-pooling is applied after every row of convolutional layers.

4.2.4 CRNN

CRNN uses a 2-layer RNN with gated recurrent units (GRU) [114] to summarise temporal patterns on the top of two-dimensional 4-layer CNNs as shown in Figure 4.1. The assumption underlying this model is that the temporal pattern can be aggregated better with RNNs then CNNs, while relying on CNNs on the input side for local feature extraction. GRU was chosen since it achieved similar performances to long short-term memory units with smaller number of parameters.

In CRNN, RNNs are used to aggregate the temporal patterns instead of, for instance, averaging the results from shorter segments as in [4] or convolution and sub-sampling as in other CNN's. In its CNN sub-structure, the sizes of convolutional layers and max-pooling layers are 3×3 and (2×2)-(3×3)-(4×4)-(4×4). This sub-sampling results in a feature map size of $N\times1\times15$ (number of feature maps×frequency×time). They are then fed into a 2-layer RNN, of which the last hidden state is connected to the output of the network.

4.2.5 Scaling networks

The models are scaled by controlling the number of parameters to be 100,000, 250,000, 0.5 million, 1M, 3M with 2% tolerance. Considering the limitation of current hardware

and the dataset size, 3M-parameter networks are presumed to provide an approximate upper bound of the structure complexity. Table 4-A summarises the details of different structures including the *layer width* (the number of feature maps or hidden units).

The widths of layers are based on [4] for `k1c2` and `k2c1`, and [15] for `k2c2`. For `CRNN`, the widths are determined based on preliminary experiments which showed the relative importance of the numbers of the feature maps of convolutional layers over the number of hidden units in RNNs.

Layer widths are changed to control the number of parameters of a network while the depths and the convolutional kernel shapes are kept constant. Therefore, the hierarchy of learned features is preserved while the numbers of the features in each hierarchical level (i.e., each layer) are changed.

4.3 Experiments

The whole training procedure is the same as introduced in the previous chapters. I use the preview clips of Million Song Dataset [103] with *last.fm* tags and train the networks to predict the top-50 tag.

The models are built with Keras [115] and Theano [116]. I use ADAM for learning rate control [40] and binary cross-entropy as a loss function. The reported performance is measured on test set and by AUC-ROC (Area Under Receiver Operating Characteristic Curve) given that tagging is a multi-label classification. The experiment was performed on NVIDIA Titan X (Pascal) GPU.

I use early-stopping for all structures – the training is stopped if there is no improvement of AUC on the validation set while iterating the whole training data once.

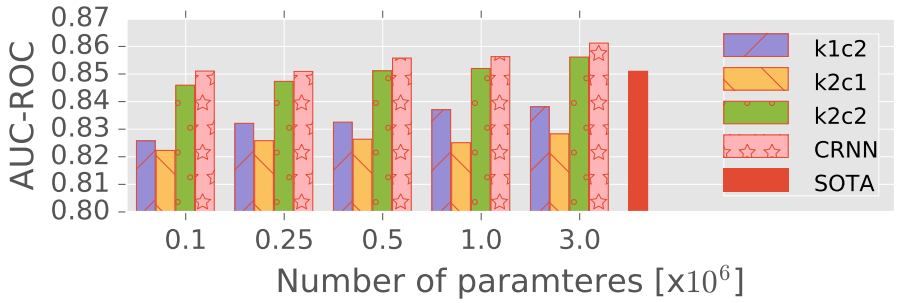

Figure 4.2: AUCs for the three structures with $\{0.1, 0.25, 0.5, 1.0, 3.0\}\times10^6$ parameters. The AUC of SOTA (state-of-the-art) is .851 [15]. SOTA result was obtained with a structure similar to `k2c2`, but with different number of layers and training scheme.

4.3.1 Memory-controlled experiment

Figure 4.2 shows the AUCs for each network against the number of parameters. With the same number of parameters, the ranking of AUC is `CRNN` > `k2c2` > `k1c2` >`k2c1`. This indicates that `CRNN` can be preferred when the bottleneck is memory usage.

`CRNN` outperforms `k2c2` in all cases. Because they share 2D-convolutional layers, this difference is probably a consequence of the difference in RNNs and CNNs the ability to summarise the features over time. This may indicate that learning a global structure is more important than focusing on local structures for summarisation. One may focus on the different layer widths of two structures – because recurrent layers use fewer parameters than convolutional layers, `CRNN` has wider convolutional layers than `k2c2` with the same number of parameters. However, even `CRNN` with narrower layer widths (0.1M parameters) shows better performance than `k2c2` with wider widths (0.25M parameters).

`k2c2` shows higher AUCs than `k2c1` and `k1c2` in all cases. This shows that the model of `k2c2`, which encodes local invariance and captures local time-frequency relationships, is more effective than the others, which ignores local frequency relationships. `k2c2` also uses parameters efficiently with its fully-convolutional structure, while `k2c1` and `k1c2` allocate only a small proportion of the parameters to the feature extraction stage. For example, in `k1c2` with 0.5M parameters, only 13% of the parameters are used by convolutional

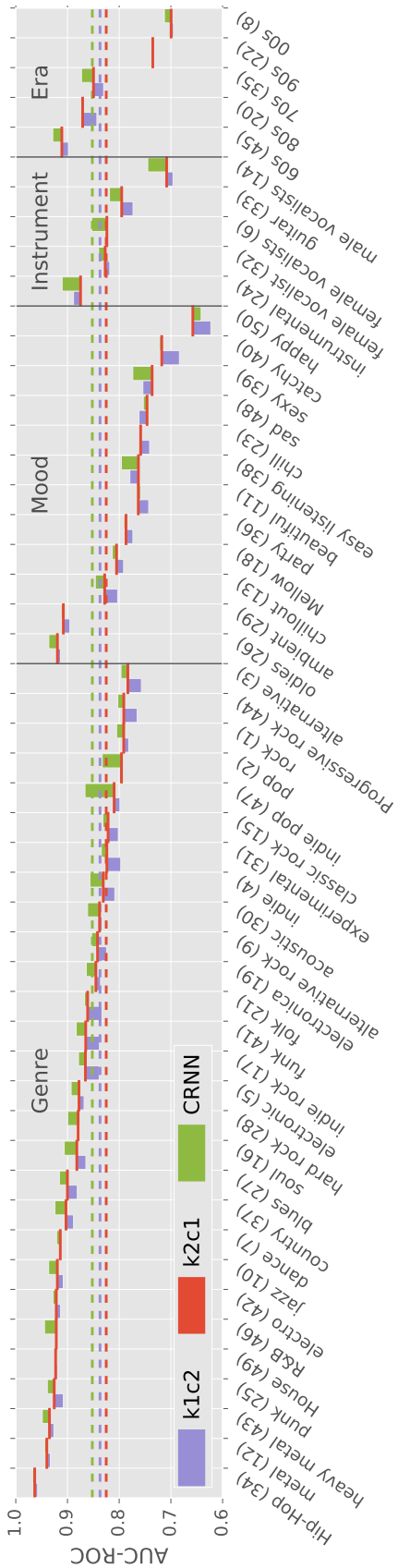

Figure 4.3: AUCs of 1M-parameter structures. *i*) The average AUCs over all samples are plotted with dashed lines. *ii*) AUC of each tag is plotted using a bar chart and line. For each tag, red line indicates the score of k2c1 which is used as a baseline of bar charts for k1c2 (blue) and CRNN (green). In other words, blue and green bar heights represent the performance gaps, k2c1-k1c2 and CRNN-k2c1, respectively. *iii*) Tags are grouped by categories (genre/mood/instrument/era) and sorted by the score of k2c1. *iv*) The number in parentheses after each tag indicates that tag's popularity ranking in the dataset.

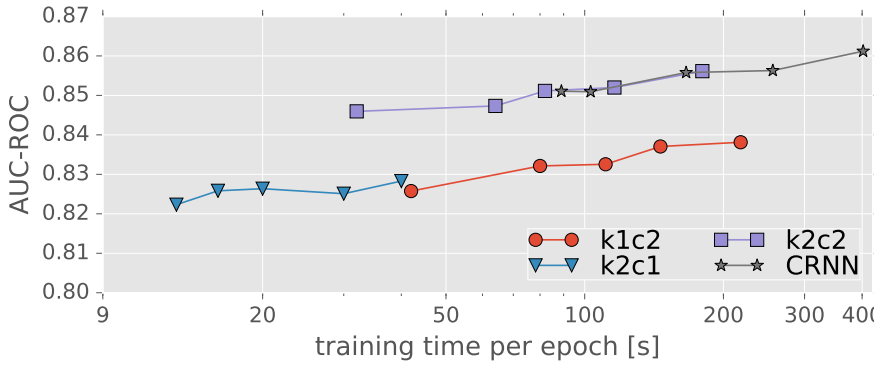

Figure 4.4: AUCs of the structures in training time - AUC plane. Each plot represents four different parameters, $\{0.1, 0.25, 0.5, 1.0, 3.0\} \times 10^6$, from left to right.

layers while the rest, 87%, are used by the fully-connected layers.

`k2c2` structures (>0.5M parameters) shows better performances than a similar but vastly larger structure in [15], which is shown as state of the art (SOTA) in Figure 4.2. This may be due to the reduction in the number of feature maps removes redundancy.

The flexibility of `k1c2` may contribute the performance improvement over `k2c1`. In `k2c1`, the *tall* 2-dimensional kernels in the first layer of `k2c1` compress the information of the whole frequency-axis pattern into each feature map. The following kernels then deal with this compressed representation with temporal convolutional and pooling. On the other hands, in `k1c2`, 1-dimensional kernels are shared over time and frequency axis until the end of convolutional layers. In other words, it gradually compresses the information in time axis first, while preserving the frequency-axis pattern.

4.3.2 Computation-controlled comparison

I further investigate the computational complexity of each structure. The computational complexity is directly related to the training and prediction time and varies depending not only on the number of parameters but also on the structure. The wall-clock training times for 2500 samples are summarised in Table 4-A and plotted in Figure 4.4.

The input compression in `k2c1` results in a fast computation, making it merely overlaps in time with other structures. The time consumptions of the other structures range in an overlapping region.

Overall, with similar training time, `k2c2` and `CRNN` show the best performance. This result indicates that either `k2c2` or `CRNN` can be used depending on the target time budget.

With the same number of parameters, the ranking of training speed is always `k2c1` > `k2c2` > `k1c2` > `CRNN`. There seem two factors that affect this ranking. **First**, among CNN structures, the sizes of feature maps are the most critical since the number of convolution operations is in proportion to the sizes. `k2c1` reduces the size of feature map in the first convolutional layer, where the whole frequency bins are compressed into one. `k2c2` reduces the sizes of feature maps in both axes and is faster than `k1c2` which reduces the sizes only in the temporal axis. **Second**, the difference between `CRNN` and CNN structures arises from the negative correlation of speed and the depth of networks. The effective depth of `CRNN` structure is up to 20 (15 time steps in RNN and 5 convolutional layers), introducing heavier computation than the other CNN structures.

4.3.3 Tag-wise performance

Figure 4.3 visualises the AUC score of each tag of 1M-parameter structures. Each tag is categorised as one of genres, moods, instruments and eras, and sorted by AUC within its category. Under this categorisation, music tagging task can be considered as a multiple-task problem equivalent to four classification tasks with these four categories.

The `CRNN` outperforms `k2c1` for 44 tags, and `k2c1` outperforms `k1c2` for 48 out of 50 tags. From the multiple-task classification perspective, this result indicates that a structure that outperforms in one of the four tasks may perform best in the other tasks as well.

Although the dataset is imbalanced, the tag popularity (number of occurrence of

each tag) is not correlated to the performance. Spearman rank correlation between tag popularity and the ranking of AUC scores of all tags is 0.077. It means that the networks effectively learn features that can be shared to predict different tags.

For further discussion on the tag-wise performance, please refer to Chapter 3 where it is analysed with respect to label noise of the dataset.

4.4 Summary

I proposed a convolutional recurrent neural network (CRNN) for music tagging. In the experiment, I controlled the size of the networks by varying the numbers of parameters to for memory-controlled and computation-controlled comparison. The experiments revealed that 2D convolution with 2d kernels (`k2c2`) and `CRNN` perform comparably to each other with a modest number of parameters. With a very small or large number of parameters, we observed a trade-off between speed and memory. The computation of `k2c2` is faster than that of `CRNN` across all parameter settings, while the `CRNN` tends to outperform it with the same number of parameters. It is worth mentioning that the approach to scale and the criteria to evaluate the models in this chapter are generally applicable regardless of the domain.

This chapter was about designing a neural network structure, which is based on the characteristic of the input, the music signal. As a music researcher, another interesting question is how can (or should) we prepare the input data so that the neural networks can extract the information better and easier. That will be discussed in the following chapter.

m

Chapter 5

The Effects of Audio Input Pre-processing

Abstract

In this chapter, I introduce my empirical analysis on audio input preprocessing for deep neural network-based music tagger. This is one of the topics that researchers in music domain would be particularly interested in, as well as an important topic to build an optimised music tagger, or classifier in general, in the real world.

A set of experiments is performed to compare the performances when varying the audio input preprocessing methods. The results show that decibel-scaling of the magnitude is critical while other methods do not result in substantial performance variance.

The work in this chapter is closely related to my EUSIPCO 2018 submission, "A Comparison of Audio Signal Preprocessing Methods for Deep Neural Networks on Music Tagging" [10].

5.1 Introduction

MIR researches that use deep learning techniques commonly focus on optimising the hyperparameters which specify the network structure. Conversely, the audio preprocessing stage is often decided on using heuristics without being subject to optimisation.

Although neural networks are known to be universal function approximators [117], training efficiency and performance may vary significantly with not only different training methods but also generic techniques including preprocessing the input data [118]. In other words, a neural network can *represent* any function but it does not mean it can always *learn* any function in practice. Optimising learning rate, for example, does not change the shape of loss function but help to converge to the better optima. In developing a deep learning-based system, both empirical decisions and domain knowledge are crucial because they can help to find a better optima. Especially, choosing between various preprocessing methods can be seen as a non-differentiable choice function, which cannot

be optimised using gradient-based learning methods.

To consider examples in prior works, mel-spectrograms have been preferred over short-time Fourier transform in many tasks [5] because it was considered to represent enough information about many problems despite its smaller size. As another example, when a time-frequency representation magnitude $\mathbf{X} \in \mathbb{R}_{\geq 0}^{N \times M}$ is given, one of the most common preprocessing approaches is to apply logarithmic compression, i.e., $\log(\mathbf{X} + \alpha)$ where α can be arbitrary constants such as very small number (e.g. 10^{-7}) or 1. However, the performances of these methods are not usually compared.

In this chapter, I focus on audio preprocessing strategies for deep convolutional neural networks for music tagging. By assessing various preprocessing strategies and providing empirical results, I aim at demystifying the effects of audio preprocessing on network performance. This will help researchers in designing deep learning systems for music research.

5.2 Experiments and discussion

A representative network structure needs to be defined to compare the effects of audio preprocessing. Compact-convnet (Section 2.4), a convnet with 2D kernels and 2D convolution axes was chosen. This showed a good performance with efficient training in Chapter 4 [8] under the name of *k2c2*, indicating 2D kernels and convolution axes. As illustrated in Figure 5.1, homogeneous 2D (3×3) convolutional kernels are used in every convolutional layer. The input has a single channel, 96 mel bins, and 1,360 temporal frames (1, 96, 1360).

For the training of music tagger, in this chapter, I used the MSD with preview audio clips as introduced previously. To remind the details, the training data are 30-60s stereo mp3 files with a sampling rate of 22,050Hz and 64 kbps constant bit-rate encoding. For efficient training in the experiments, I downmix and downsample the signals to 12kHz

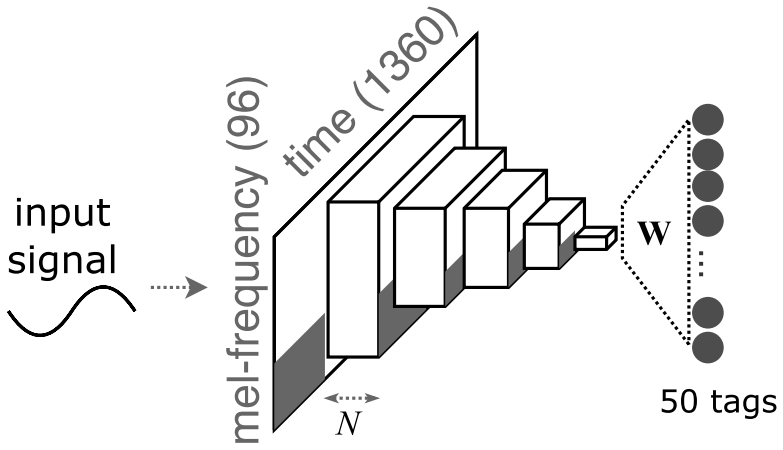

Figure 5.1: Network structure of the 5-layer convnet. N refers to the number of feature maps (which is set to 32 for all layers in this chapter) while $\mathbf{W}$ refers to the weights matrix of the fully-connected output layer.)

after decoding and trimming the audio duration to 29-second to ensure equal-sized input signals. The short-time Fourier transform and mel-spectrogram are computed using a hop size of 256 samples (21.3 ms) with a 512 point DFT aggregated to yield 96 mel bins per frame. The preprocessing is performed using Librosa [119] and *Kapre* [9]. The training scheme is identical to the previous chapters. The binary cross-entropy function is used as a loss function. For the acceleration of stochastic gradient descent, I use adaptive optimisation based on Adam [40].

5.2.1 Variance by different initialisation

In deep learning, using K-fold cross-validation is not a standard practice for two reasons. First, with large enough data and a good split of train, validation, and test sets, the model can be trained with small variance. Second, the cost of hyperparameter search is very high and it makes repeating experiments too expensive in practice. For these reasons, I do not cross-validate the convnet in this study. Instead, I present the results of repeated experiments with fixed network and training hyperparameters such as training example sequences and batch size. This experiment, therefore, measures the variance of the

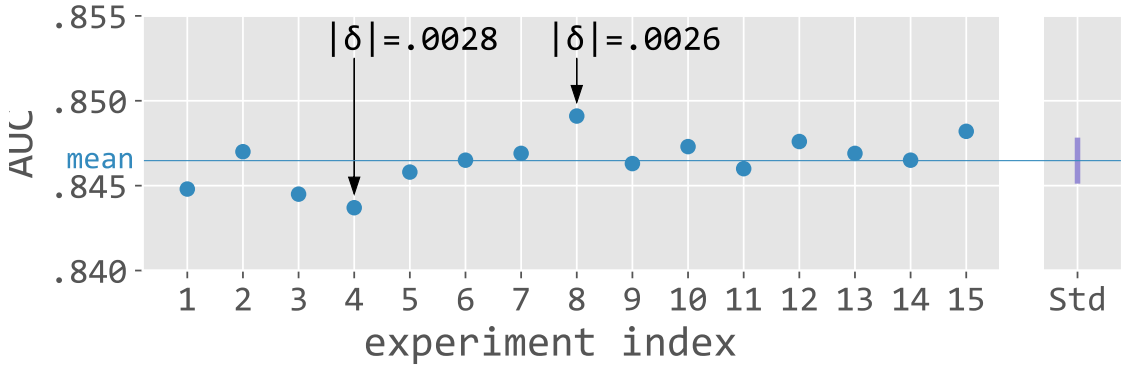

Figure 5.2: Performances and their mean (left), as well as the 95% confidence interval (CI). The two deltas on the plot indicate the difference between the average AUC and the scores of experiments 4 and 8.

model introduced by different weight initialisations of the convolutional layers. For this, a normal distribution is used following He et al. [120], which has been shown to yield a stable training procedure.

The results are summarised in Figure 5.2. This shows the AUC scores of 15 repeated experiments on the left as well as their standard deviation on the right. Small standard deviation indicates that we can obtain a reliable, precise score by repeating the same experiments for a sufficient number of times. The two largest differences observed between the average AUC score and that of experiment 4 and 8 (AUC differences of 0.0028 and 0.0026 respectively) indicate that we may obtain up to ~ 0.005 AUC difference among experiment instances. Based on this, I assume that an AUC difference of < 0.005 is non-significant in this chapter.

5.2.2 Time-frequency representations

STFT and mel-spectrogram have been the most popular input representations for music classification [5]. Although sample-based deep learning methods have been introduced, 2-dimensional representations would be still useful in the near future for efficient training. Mel-spectrograms provide an efficient and perceptually relevant representation compared

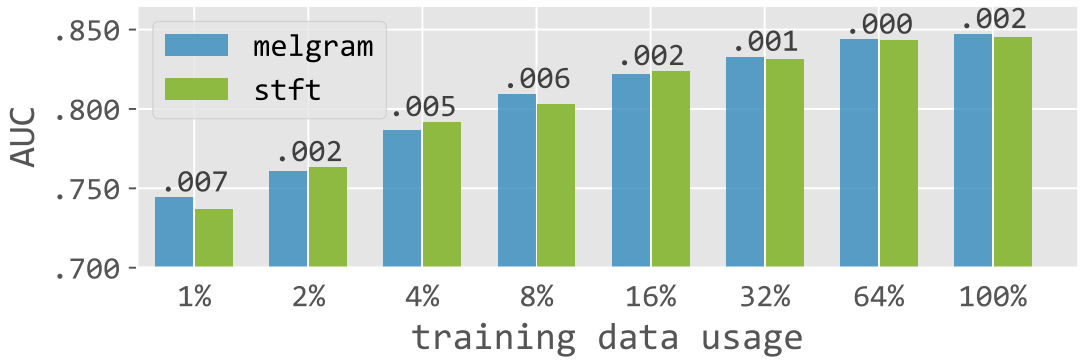

Figure 5.3: Performances of predictions with mel-spectrogram and STFT with varying training data sizes. The numbers above bars indicate the absolute performance differences between mel-spectrograms and STFTs.

to STFT [49] and have been shown to perform well in various tasks [4, 15, 54–56]. However, an STFT is closer to the original signal and neural networks may be able to learn a representation that is more optimal to the given task than mel-spectrograms. This requires large amounts of training data, however, as reported in [15] where using mel-spectrograms outperformed STFT with a smaller dataset.

Figure 5.3 shows the AUC scores obtained using log(STFT) vs. log(mel-spectrogram) while varying the size of the utilised training data. Although there are small differences on AUC scores up to 0.007, neither of them outperforms the other, especially when enough data is provided. This rebuts the results reported in [15] because mel-spectrograms did not have a clear advantage here, even with a small training data size. This may be explained by the higher frequency resolution of the STFT representation used and summarised as follows.

- STFT in [15]: 6000/129=46.5 Hz (256-point FFT with 12 kHz sampling rate)
- STFT in this work: 6000/257=23.3 Hz (512-point FFT with 12 kHz sampling rate)
- Mel-spectrogram in [15] and this work: 35.9 Hz for frequency $<$ 1 kHz (96 mel-bins and by [121] and [119])

In [15], the frequency resolution of the STFT was lower than that of the mel-spectrogram to enable comparing them with a similar number of frequency bins. On the contrary, STFT of higher frequency resolution is used in this experiment. Nevertheless, it is found to be only as good as mel-spectrogram in terms of performance. This means overall that, in practice, mel-spectrogram may be preferred since its smaller size leads to reduced computation in training and prediction. However, this may be because of the limit of the used networks which does not take advantage of finer input rather than because of the nature of the task. For example, detecting vocal component (to tag 'instrumental') could use finer frequency structure of the consonant part, but only if the network is designed to capture them effectively. The figure also illustrates how much the data size affects the performance. Exponentially increasing data size merely results in a linear AUC improvement. AUC starts to converge at `64%` and `100%`.

5.2.3 Analysis of scaling effects and frequency-axis weights

Lastly, I discuss the effects of magnitude manipulation. Preliminary experiments suggested that there might be two independent aspects to investigate; *i)* frequency-axis weights and *ii)* magnitude scaling of each item in the training set. Examples of frequency-axis weights are illustrated in Figure 5.4, where different weighting schemes are plotted. The experiment is designed to isolate these two effects. I tested two input representations *log-mel-spectrogram* (mel-spectrogram in decibel scale) vs. *mel-spectrogram*, with three frequency weighting schemes *per-frequency*, *A-weighting* and *bypass*, as well as two scaling methods $\times 10$ (`on`) and $\times 1$ (`off`), yielding $2 \times 3 \times 2 = 12$ configurations in total.

I summarise the mechanism of each block as follows. First, there are three frequency weights schemes.

- `per-frequency stdd`: This is often called spectral whitening. I compute means and standard deviations across time, i.e., per-frequency and standardise each frequency band using these values. The average frequency response becomes flat (equalised).

This method has been used in the literature for tagging [15], singing voice detection [88] and transcription [58].

- `A-weighted`: I apply the international standard IEC 61672:2003 A-weighting curve, which approximates human perception of loudness as a function of frequency.
- `Bypass`: Do not apply any processing, i.e., $f : \mathbf{X} \to \mathbf{X}$

There are two other blocks which are mutually independent and also independent to frequency weights schemes.

- `per-sample stdd`: Excerpt-wise normalisation with its overall mean and standard deviation, i.e., using statistics across time and frequency of each spectrogram.
- `×10 scaler`: Multiply the input spectrogram by 10, i.e.,$f : \mathbf{X} \to 10\mathbf{X}$.

In the following sections, I discuss result on frequency weighting and scaling effects respectively.

5.2.3.1 Frequency weighting

This process is related to loudness, i.e., human perception of sound energy [49], which is a function of frequency. The sensitivity of the human auditory system drops substantially below a few hundred Hz[1], hence music signals are often produced to exhibit higher physical energy in the lower frequency range in order to be perceptually balanced. This is illustrated in Figure 5.4, where uncompensated average energy measurements corresponding to the *Bypass* curve (shown in green) yield a peak at low frequencies. This imbalance affects neural network activations in the first layer which may influence performance. To assess this effect, I tested three frequency weighting approaches. Their typical profiles are shown in Figure 5.4. In all three strategies, excerpt-wise standardisation is used to alleviate scaling effects (see Section 5.2.3.2).

[1] See equal-loudness contours e.g. in ISO 226:2003.

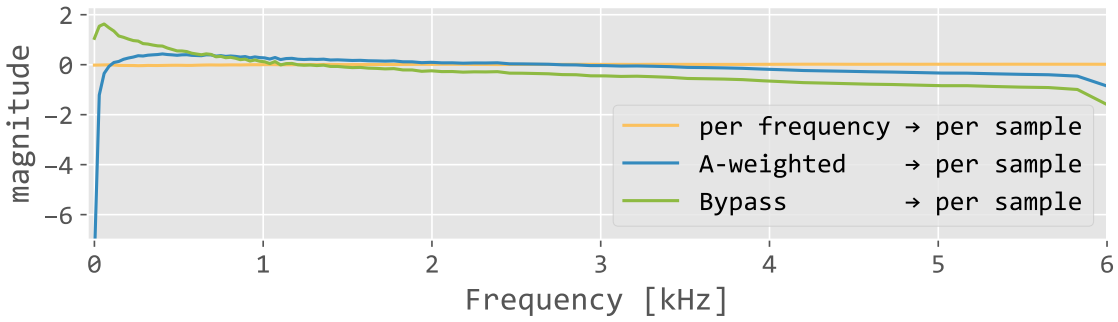

Figure 5.4: Average frequency magnitude of randomly selected 200 excerpts with three frequency-axis normalisation. A per-sample (excerpt) standardisation follows to remove the effect of different overall average magnitude.

The test results show that networks using the three strategies all achieve similar AUC scores. The results are illustrated in Figure 5.5 where the colours indicate normalization schemes.[2] The performance differences *within* four groups {1, 1s, 2, 2s} shown in Figure 5.5 are small and none of them are significantly different the others, which indicates they do not produce a significant effect on the training. The curves in Figure 5.4 show the average input magnitudes over frequency. These offsets change along frequency, but the change does not seem large enough to corrupt the local patterns due to the locality of convnets, and therefore the network is learning useful representations without significant performance differences within each group.

5.2.3.2 Analysis of scaling effects

For a number of reasons discussed below, we may assume a performance increase if we scale the overall input magnitudes. During training using gradient descent, the gradient of error with respect to weights $\frac{\partial E}{\partial W}$ is proportional to $\frac{\partial}{\partial W} f(W^\top X)$ where f is the activation function. This means that the learning rate of a layer is proportional to the magnitude of input X. In particular, the first layer usually receives the weakest error backpropagation, hence scaling of the input may affect the overall performance.

[2] Additionally, {1, 2 vs. 1s, 2s} compares scaling effect (see Section 5.2.3.2) and {1, 1s} vs. {2, 2s} compares the log-compression effect (see Section 5.2.4).

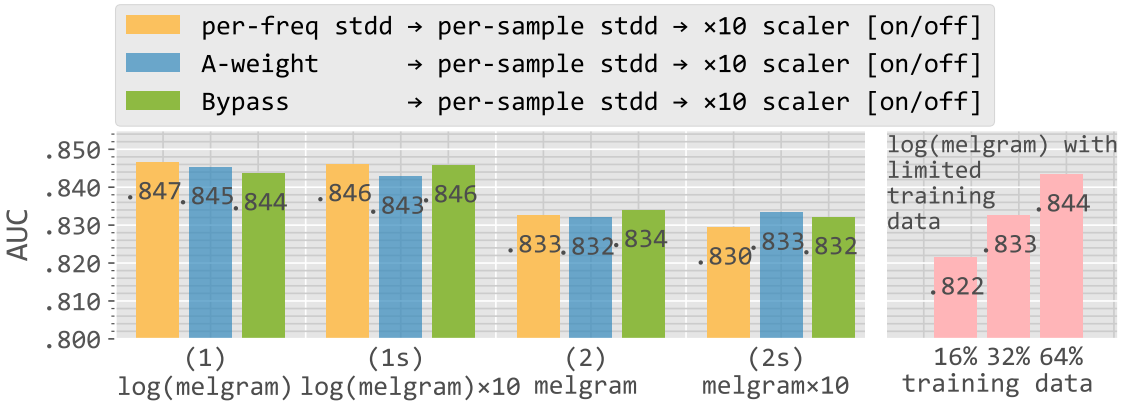

Figure 5.5: Performance comparisons of different preprocessing procedures. From left to right, four groups of scores are from different magnitude processing (mel-spectrogram in decibel scale and linear scale), with additional ×10 scaler turned on/off. In each group, yellow/blue/green bars indicate different frequency-axis processing as annotated in the legend. Logarithmic compression is done before other preprocessing methods are applied.

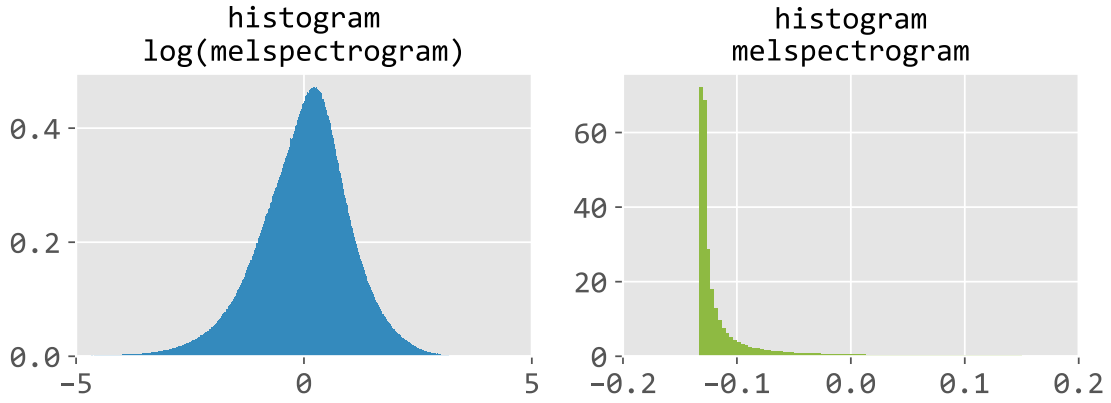

Figure 5.6: histograms of the magnitude of mel-spectrogram time-frequency bins with (left) and without (right) logarithmic compression. The number of bins are 100 and both are normalised, i.e., $\sum_{i=1}^{100} 0.01 \times y_i = 1$. Log compression significantly affects the histogram, making the distribution Gaussian (left), otherwise extremely skewed (right). This is after standardisation and based on randomly selected 100 tracks from the training set.

I tested the effect of this with the results shown in Figure 5.5. To this end, consider comparing the matching colour bars of {`1` vs. `1s`} and {`2` vs. `2s`}. Here, the scaling factor is set to 10 for illustration, however many possible values <100 were tested and

showed similar results. In summary, this hypothesis is rebutted as scaling did not affect the performance. The analysis of trained weights revealed that different magnitudes of the input only affects the bias of the first convolutional layer. Training with scaling set to $\times 10$ results in about 3.4 times larger mean absolute value of the biases in the first layer. This may be due to batch normalization [99] which compensates for the different magnitudes by normalising the activations of convolutional layers.

5.2.4 Log-scaling of magnitudes

Lastly, I discuss how logarithmic compression of magnitudes, i.e. decibel scaling, affects performance. This is considered standard preprocessing in music information retrieval. The procedure is motivated by the human perception of loudness [49] which has a logarithmic relationship with the physical energy of sound. Although learning a logarithmic function may be a trivial task for neural networks, it can be difficult to implicitly learn an optimal nonlinear compression when it is embedded in a complicated task. For the same reason, a nonlinear compression was also shown to affect the performance in visual image recognition using neural networks [122]. Figure 5.6 compares the histograms of the magnitudes of time-frequency bins after zero-mean unit-variance standardisation. On the left, a logarithmically compressed mel-spectrogram shows an approximately Gaussian distribution without any extreme values. Meanwhile, the bins of linear mel-spectrogram on the right is extremely condensed in a very small range while they range in wider region overall. This means the network should be trained with higher numerical precision to the input, hence more vulnerable to noise.

As a result, log-compressed mel-spectrograms always outperform the linear versions as shown in Figure 5.5, where matching colour bars should be compared across within {1 vs. 2} and {1s vs. 2s}[3]. The additional work introduced by *not* using log-compression can be roughly estimated by comparing these scores to those networks when the training data size is limited (shown in pink on the right of Figure 5.5). While this also depends

[3]Log-compressed STFT also outperformed linear STFT in the initial experiments.

on other configurations of the task, seemingly twice the data is required to compensate for the disadvantage of not using a log-compressed representation. This is a significant difference as the non-trivial burden for collecting labelled data and training time. It is noteworthy that this specific 'doubling' relationship would differ by the task and data, e.g. [123].

5.3 Summary

In this chapter, I have shown that some of the input preprocessing methods can affect the performance of neural networks for music tagging. I quantify this in terms of the size of the training data required to achieve similar performances. Among several pre-processing techniques tested in this study, only logarithmic scaling of the magnitude resulted in significant improvement. In other words, the network was resilient to most modifications of the input data except logarithmic compression of magnitudes in various time-frequency representations. Considering the diverse nature of music tags that cover genre, mood, and instruments, our results provide practical advice applicable in many similar machine-listening problems, e.g., music genre classification or mood prediction. Since the introduced experiments are based on pair-wise comparison instead of a grid-search, there is a possibility in theory that a combination of non-optimal choices would result in better performance. However, the parameters that I tested in this chapter do not seem to interact each other, hence it is unlikely that such a combination exists. In a long term, end-to-end approaches would get more attention and become a computationally viable option, but by then, a mel-spectrogram with log-magnitude would be at least a good starting point for a new task/dataset in MIR.

A supervised learning system learns a suitable mapper $f : X \rightarrow Y$. So far, as a practitioner of neural networks in music research, I discussed all the three components – music signal preprocessing (X), the network structures (f), and the properties of the label (Y). In the following chapter, I will introduce how can we *re-use* a trained network

as a feature extractor that can be generally used in music classification and regression tasks.

Chapter 6

Transfer Learning for Music Informatics Tasks

Abstract

This chapter introduces my work on 'transfer learning', a method to re-use a trained network on other tasks and/or datasets. In general, transfer learning is gaining attention in the research fields because not every task/every researcher are provided with a large dataset to train a network, which is true in many MIR tasks.

In this chapter, I propose to use a *pre-trained convnet feature*. I show how this convnet feature can serve as a general-purpose music representation. In the experiments, a convnet is trained for music tagging and then transferred to other music-related classification and regression tasks. The convnet feature outperforms the baseline MFCC feature in all the considered tasks and several previous approaches that are aggregating low- and high-level music features.

This chapter is based on my ISMIR 2017 paper, "Transfer Learning for Music Classification and Regression Tasks" [11] which was awarded the best paper.

6.1 Introduction

Transfer learning is often defined as *re-using parameters* that are trained on a *source* task for a *target* task, aiming to transfer knowledge between the domains. A common motivation for transfer learning is the lack of sufficient training data in the target task. When using a neural network, by transferring pre-trained weights, the number of trainable parameters in the target-task model can be significantly reduced, enabling effective learning with a smaller dataset.

A popular example of transfer learning is semantic image segmentation in computer vision, where the network utilises rich information, such as basic shapes or prototypical templates of objects, that were captured when trained for image classification [124]. Another example is pre-trained word embeddings in natural language processing. Word

embedding, a vector representation of a word, can be trained on large datasets such as Wikipedia [125] and adopted to other tasks such as sentiment analysis [126].

In the case of deep neural networks, a common approach is to freeze a once-trained network, remove the final 1 or several layers, and add a new output layer which is trained for the target task.

There have been several works on transfer learning in MIR. Hamel et al. proposed to directly learn music features using linear embedding [127] of mel-spectrogram representations and genre/similarity/tag labels [64]. Oord et al. outlines a large-scale transfer learning approach, where a multi-layer perceptron is combined with the spherical K-means algorithm [128] trained on tags and play-count data [129]. After training, the weights are transferred to perform genre classification and auto-tagging with smaller datasets. In music recommendation, Choi et al. used the weights of a convolutional neural network for feature extraction in playlist generation [130], while Liang et al. used a multi-layer perceptron for feature extraction of content-aware collaborative filtering [31].

6.2 Transfer learning for music

In this section, the proposed transfer learning approach is described. A convolutional neural network (convnet) is designed and trained for a source task, and then, the network with trained weights is used as a feature extractor for target tasks. The schematic of the proposed approach is illustrated in Figure 6.1.

6.2.1 Convolutional neural networks for music tagging

I argue that music tagging is suitable as a source task because *i)* large training data is available and *ii)* its rich label set covers various aspects of music, e.g., *genre*, *mood*, *era*, and *instrumentations*. The convnet used in this chapter is Compact-convnet which was

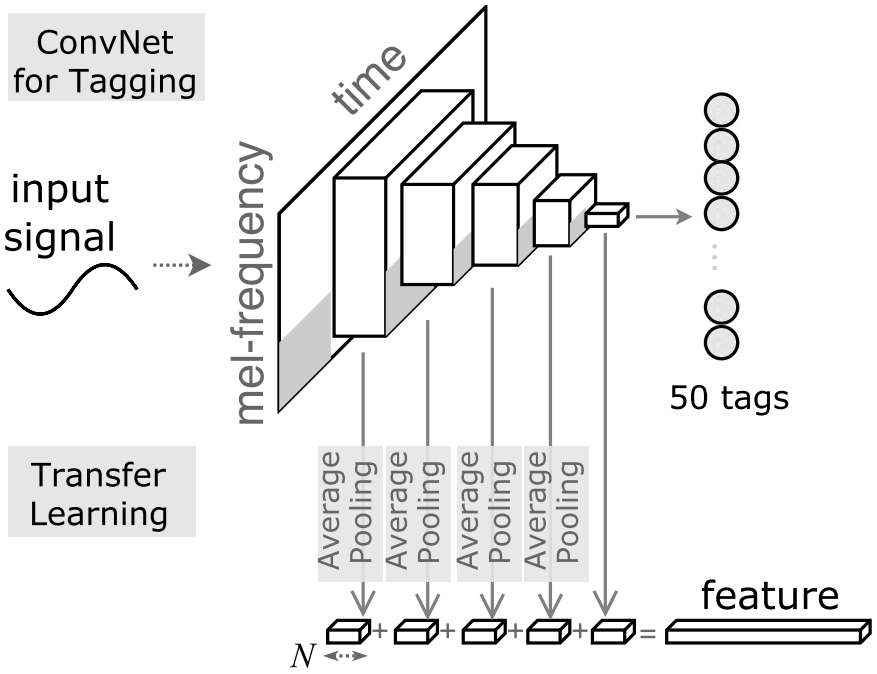

Figure 6.1: A block diagram of the training and feature extraction procedures. The convnet structure is identical to Compact-convnet (Section 2.4). After training, the feature maps from 1st–4th layers are subsampled using average pooling while the feature map of 5th layer is used as it is, since it is already scalar (size 1×1). Those 32-dimensional features are concatenated to form a *convnet feature*.

introduce in 2.4 with the best performing input preprocessing strategy in Chapter 5. To remind the details, a 96-bin mel-spectrogram ($\mathbf{X}$) is used as the input to the convnet. Its magnitude is mapped to decibel scale ($\log_{10} \mathbf{X}$). There are five layers of convolutional and sub-sampling in the convnet as shown in Figure 6.1. This convnet structure with 2-dimensional 3×3 kernels and 2-dimensional convolution.

6.2.2 Representation transfer

In this section, I explain how features are extracted from a pre-trained convolutional network. In the remainder of the chapter, this feature is referred to as *pre-trained convnet feature*, or simply *convnet feature*.

It is already well understood how deep convnets learn *hierarchical features* in visual image classification [131]. By convolution operations in the forward path, lower-level

features are used to construct higher-level features. Subsampling layers reduce the size of the feature maps while adding local invariance. In a deeper layer, as a result, the features become more invariant to (scaling/location) distortions and more relevant to the target task.

This type of hierarchy also exists when a convnet is trained for a music-related task. Visualisation and sonification of convnet features for music genre classification has shown the different levels of hierarchy in convolutional layers [13], [12]. (This is further discussed in the following chapter, Chapter 7.)

Such a hierarchy serves as a motivation for the proposed transfer learning. Relying solely on the last hidden layer may not maximally extract the knowledge from a pre-trained network. For example, low-level information such as tempo, pitch, (local) harmony or envelope can be captured in early layers, but may not be preserved in deeper layers due to the constraints that are introduced by the network structure: aggregating local information by discarding information in subsampling. For the same reason, deep scattering networks [132] and a convnet for music tagging introduced in [17] use multi-layer representations.

Based on this insight, I propose to use not only the activations of the final hidden layer but also the activations of (up to) *all* intermediate layers to find the most effective representation for each task. The final feature is generated by concatenating these features as demonstrated in Figure 6.1, where all the five layers are concatenated to serve as an example.

Given five layers, there are $\sum_{n=1}^{5} \binom{5}{n} = 31$ strategies of layer-wise combinations. In the experiment, I perform a nearly exhaustive search and report all results. I designate each strategy by the indices of layers employed. For example, a strategy named '135' refers to using a $32 \times 3 = 96$-dimensional feature vector that concatenates the first, third, and fifth layer convnet features.

During the transfer, average-pooling is used for the 1st–4th layers to reduce the size

of feature maps to 1×1 as illustrated in Figure 6.1. Averaging is chosen instead of `max` pooling because it is more suitable for summarising the global statistics of large regions, as done in the last layer in [133]. Max-pooling is often more suitable for capturing the existence of certain patterns, usually in small and local regions[1].

Lastly, there have been works suggesting random-weights (deep) neural networks including deep convnet can work well as a feature extractor [134] [135] (Not identical, but a similar approach is transferring knowledge from an irrelevant domain, e.g., visual image recognition, to music task [136].) I report these results from random convnet features and denote it as *random convnet feature*. Assessing performances of random convnet feature will help to clarify the contributions of the pre-trained knowledge transfer versus the contributions of the convnet structure and nonlinear high-dimensional transformation.

6.2.3 Classifiers and regressors of target tasks

Variants of Support vector machines (SVMs) [137, 138] are used as a classifier and regressor. SVMs work efficiently in target tasks with small training sets, and outperformed K-nearest neighbours in my work for all the tasks in a preliminary experiment. Since there are many works that use hand-written features and SVMs, using SVMs enables us to focus on comparing the performances of features.

SVMs were chosen for a number of reasons. First, with small amount of data, they often outperform many other classifiers including neural networks. Second, within the given ranges of the number of items and dimensionalities (few hundreds - thousands items and up to 128 dimensions), SVMs can scale well, being trained in a reasonable amount of time. Third, since SVMs were extremely popular in mid-2000s, there are enough number of research that are based on SVMs but with different feature sets, which makes it easy to evaluate the proposed convnet feature as a representation.

[1]Since the average is affected by zero-padding which is applied to signals that are shorter than 29 seconds, those signals are repeated to create 29-second signals. This only happens in Task 5 and 6 in the experiment.

Task	Dataset name	#clips	Metric	#classes
T1. Ballroom dance genre classification	Extended ballroom [139]	4,180	Accuracy	13
T2. Genre classification	Gtzan genre [140]	1,000	Accuracy	10
T3. Speech/music classification	Gtzan speech/music [141]	128	Accuracy	2
T4. Emotion prediction	EmoMusic (45-second) [142]	744	Coefficient of determination (r^2)	N/A (2-dimensional)
T5. Vocal/non-vocal classification	Jamendo [143]	4,086	Accuracy	2
T6. Audio event classification	Urbansound8K [144]	8,732	Accuracy	10

Table 6-A: The details of the six tasks and datasets used in the transfer learning evaluation.

6.3 Preparation

6.3.1 Source task: music tagging

The training details of source task are identical to other cases in the previous chapters, Chapter 3 and 4. In the source task, 244,224 preview clips of the Million Song Dataset [103] are used with top-50 *last.fm* tags. Binary cross-entropy is used as the loss function. The ADAM optimisation algorithm [145] is used for accelerating stochastic gradient descent. The convnet achieves 0.849 AUC-ROC score (Area Under Curve - Receiver Operating Characteristic) on the test set.

6.3.2 Target tasks

Six datasets are selected to be used in six target tasks. They are summarised in Table 6-A.

- Task 1: The Extended ballroom dataset consists of specific Ballroom dance subgenres.
- Task 2: The Gtzan genre dataset has been extremely popular, although some flaws have been found [146].
- Task 3: The dataset size is smaller than the others by an order of magnitude.

- Task 4: Emotion prediction on the arousal-valence plane. I evaluate arousal and valence separately. I trim and use the first 29-second from the 45-second signals.

- Task 5. Excerpts are subsegments from tracks with binary labels (*'vocal'* and *'non-vocal'*). Many of them are shorter than 29s. This dataset is provided for benchmarking frame-based vocal detection while I use it as a pre-segmented classification task, which may be easier than the original task.

- Task 6: This is a non-musical task. For example, the classes include *air conditioner*, *car horn*, and *dog bark*. All excerpts are shorter than 4 seconds.

6.3.3 Baseline feature and random convnet feature

As a baseline feature, the means and standard deviations of 20 Mel-Frequency Cepstral Coefficients (MFCCs), and their first and second-order derivatives are used. In this chapter, this baseline feature is called *MFCCs* or *MFCC vectors*. MFCC is chosen since it has been adopted in many music information retrieval tasks and is known to provide a robust representation. *Librosa* [119] is used for MFCC extraction and audio processing.

The random convnet feature is extracted using the identical convnet structure of the source task and after random weights initialisation with a normal distribution ('he' normalisation) [120] but without a training.

6.4 Experiments

6.4.1 Configurations

For Tasks 1-4, the experiments are done with 10-fold cross-validation using stratified splits. For Task 5, pre-defined training/validation/test sets are used. The experiment on Task 6 is done with 10-fold cross-validation without replacement to prevent using

the sub-segments from the same recordings in training and validation. The SVM hyper-parameters are optimised using grid-search based on the validation results. Kernel type-/bandwidth of radial basis function and the penalty parameter are selected from the ranges below:

- Kernel type: [*linear*, *radial*]
  - Bandwidth γ in radial basis function : $[1/2^3, 1/2^5, 1/2^7, 1/2^9, 1/2^{11}, 1/2^{13}, 1/N_f]$
- Penalty parameter C : $[0.1, 2.0, 8.0, 32.0]$

A radial basis function is $\exp(-\gamma|x - x'|^2)$, and γ and N_f refer to the radial kernel bandwidth and the dimensionality of feature vector respectively. With larger C, the penalty parameter or regularisation parameter, the loss function gives more penalty to misclassified items and vice versa. I use *Scikit-learn* [147] for these target tasks. The code for the data preparation, experiment, and visualisation are available on GitHub[2].

6.4.2 Results and discussion

Figure 6.2 shows a summary of the results. The scores of the *i)* best performing convnet feature, *ii)* concatenating '12345'[3] convnet feature and MFCCs, *iii)* MFCC feature, and *iv)* state-of-the-art algorithms for all the tasks.

In all the six tasks, the majority of convnet features outperforms the baseline feature. Concatenating MFCCs with '12345' convnet feature usually does not show improvement over a pure convnet feature except in Task 6, audio event classification. Although the reported state-of-the-art is typically better, almost all methods rely on musical knowledge and hand-crafted features, yet my features perform competitively. An in-depth look at each task is therefore useful to provide insight.

[2] https://github.com/keunwoochoi/transfer_learning_music

[3] Again, '12345' refers to the convnet feature that is concatenated from 1st–5th layers. For another example, '135' means concatenating the features from first, third, and fifth layers.

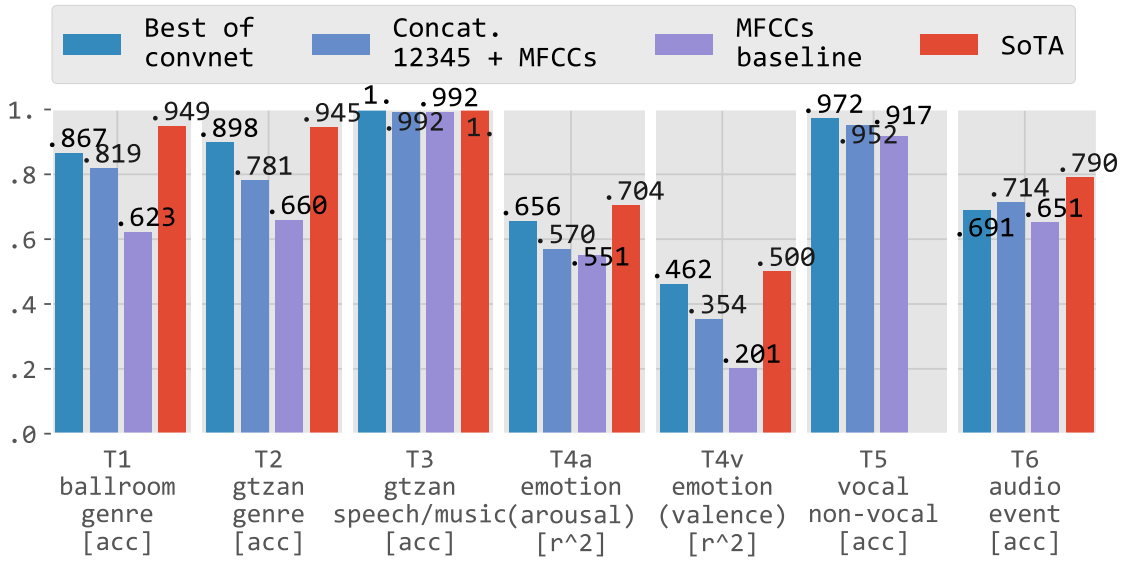

Figure 6.2: Summary of performances of the convnet feature (blue), MFCCs (purple), and state-of-the-art (red) for Task 1-6 (State-of-the-art of Task 5 does not exist).

In the following subsections, the details of each task are discussed with more results presented from (almost) exhaustive combinations of convnet features as well as random convnet features at all layers. For example, in Figure 6.3, the scores of 28 different convnet feature combinations are shown with blue bars. The narrow, grey bars next to the blue bars indicate the scores of random convnet features. The other three bars on the right represent the scores of the concatenation of '12345' + MFCC feature, MFCC feature, and the reported state-of-the-art methods respectively. The rankings within the convnet feature combinations are also shown *in* the bars where top-7 and lower-7 are highlighted.

I only briefly discuss the results of random convnet features here. The best performing random convnet features do not outperform the best-performing convnet features in any task. In most of the combinations, convnet features outperformed the corresponding random convnet features, although there are few exceptions. However, random convnet features also achieved comparable or even better scores than MFCCs, indicating *i)* a significant part of the strength of convnet features comes from the network structure itself, and *ii)* random convnet features can be useful especially if there is not a suitable

source task.

6.4.2.1 Task 1. Ballroom genre classification

Figure 6.3 shows the performances of different features for Ballroom dance classification. The highest score is achieved using the convnet feature '123' with 86.7% of accuracy. The convnet feature shows good performances, even outperforming some previous works that explicitly use rhythmic features.

The result clearly shows that low-level features are crucial in this task. All of the top-7 strategies of convnet feature include the *second* layer, and 6/7 of them include the *first* layer. On the other hand, the lower-7 are ['5', '4', '3', '45', '35', '2', '25'], none of which includes the first layer. Even '1' achieves a reasonable performance (73.8%).

The importance of low-level features is also supported by known properties of this task. The ballroom genre labels are closely related to rhythmic patterns and tempo [139] [148]. However, there is no label directly related to tempo in the source task. Moreover, deep layers in the proposed structure are conjectured to be mostly invariant to tempo. As a result, high-level features from the fourth and fifth layers poorly contribute to the task relative to those from the first, second, and third layers.

The state-of-the-art algorithm which is also the only algorithm that used the same dataset due to its recent release uses *2D scale transform*, an alternative representation of music signals for rhythm-related tasks [149], and reports 94.9% of weighted average recall. For additional comparisons, there are several works that use the Ballroom dataset [150]. This has 8 classes and it is smaller in size than the Extended Ballroom dataset (13 classes). Laykartsis and Lerch [151] combines beat histogram and timbre features to achieve 76.7%. Periodicity analysis with SVM classifier in Gkiokas et al. [152] respectively shows 88.9%/85.6 - 90.7%, before and after feature selection.

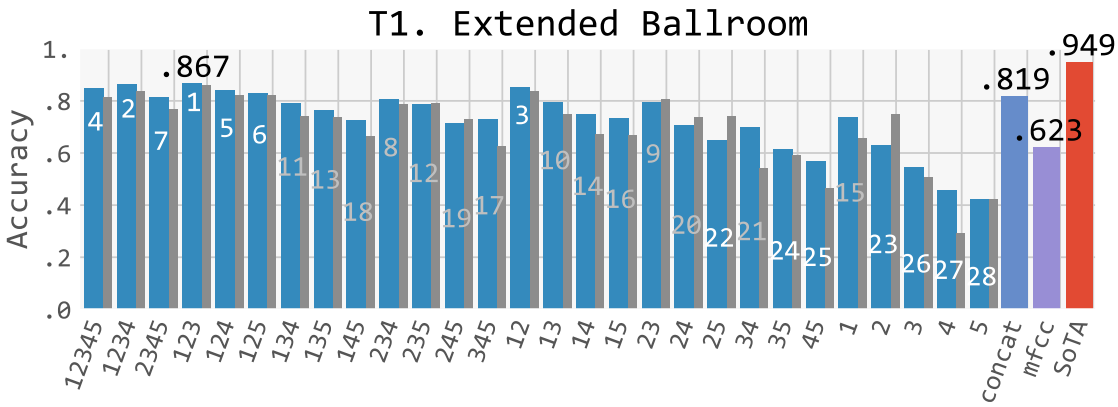

Figure 6.3: Performances of Task 1 - Ballroom dance genre classification of convnet features (with random convnet features in grey), MFCCs, and the reported state-of-the-art method. (Note the exception that the state-of-the-art score is reported in weighted average recall.)

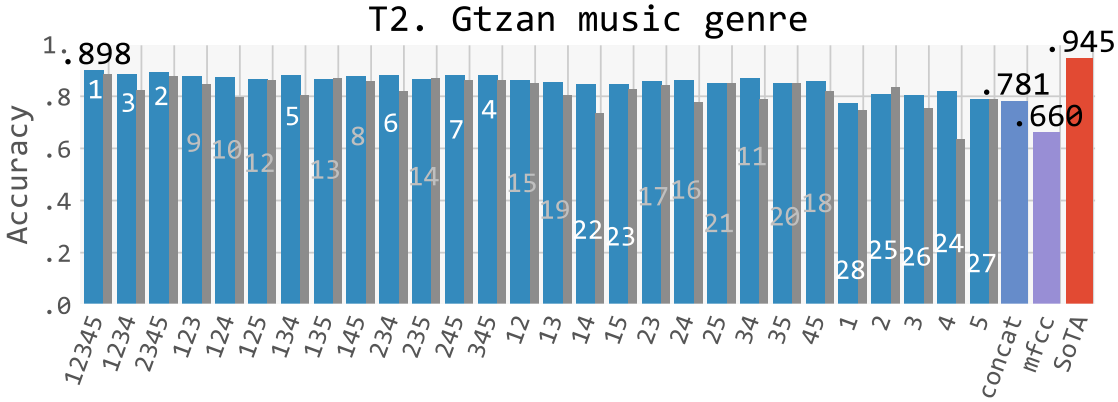

Figure 6.4: Performances of Task 2 - Gtzan music genre classification of convnet features (with random convnet features in grey), MFCCs, and the reported state-of-the-art method.

6.4.2.2 Task 2. Gtzan music genre classification

Figure 6.4 shows the performances on Gtzan music genre classification. The best-performing convnet feature shows 89.8% while the concatenated feature and MFCCs respectively show only 78.1% and 66.0% of accuracy. Although there are methods that report accuracies higher than 94.5%, I set 94.5% as the state-of-the-art score following the dataset analysis in [146], which shows that the perfect score cannot surpass 94.5% considering the noise in the Gtzan dataset.

Among a significant number of works that use the Gtzan music genre dataset, I

describe four methods in more detail. Three of them use an SVM classifier, which enables us to focus on the comparison with the proposed feature. Arabi and Lu [153] is most similar to the proposed convnet features in a way that it combines low-level and high-level features and shows a similar performance. Beniya et al. [154] and Huang et al. [155] report the performances with many low-level features before and after applying feature selection algorithms. Only the latter outperforms the proposed method and only after feature selection.

- Arabi and Lu [153] uses not only low-level features such as {spectral centroid/flatness/roll-off/flux}, but also high-level musical features such as {beat, chord distribution and chord progressions}. The best combination of the features shows 90.79% of accuracy.

- Beniya et al. [154] uses a particularly rich set of statistics such as {mean, standard deviation, skewness, kurtosis, covariance} of many low-level features including {RMS energy, attack, tempo, spectral features, zero-crossing, MFCC, dMFCC, ddMFCC, chromagram peak and centroid}. The feature vector dimensionality is reduced by MRMR (max-relevance and min-redundancy) [156] to obtain the highest classification accuracy of 87.9%.

- Huang et al. [155] adopts another feature selection algorithm, self-adaptive harmony search [157]. The method uses statistics such as {mean, standard deviation} of many features including {energy , pitch, and timbral features} and their derivatives. The original 256-dimensional feature achieved 84.3% of accuracy which increases to 92.2% and 97.2% after feature selection.

- Reusing AlexNet [2], a pre-trained convnet for visual image recognition achieved 78% of accuracy [136].

In summary, the convnet feature achieves better performance than many approaches which use extensive music feature sets without feature selection as well as some of the approaches with feature selection. For this task, it turns out that combining features

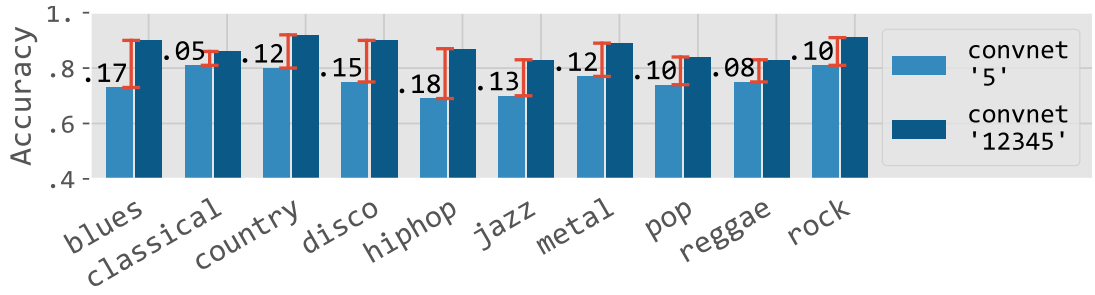

Figure 6.5: Comparison of per-label results of two convnet feature strategies, '12345' and '5' for Gtzan music genre classification. Numbers denote the differences of scores.

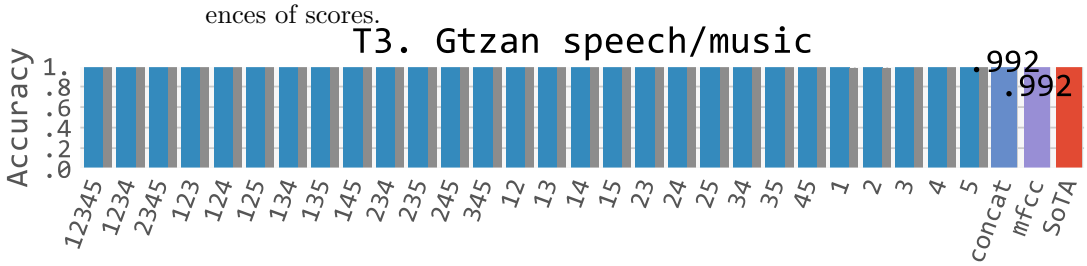

Figure 6.6: Performances of Task 3 - Speech/music classification of convnet features (with random convnet features in grey), MFCCs, and the reported state-of-the-art method. All scores of convnet features and SoTA are 1.0 and omitted in the plot.

from all layers is the best strategy. In the results, '12345', '2345', and '1234' are three best configurations, and all of the top-7 scores are from those strategies that use more than three layers. On the contrary, all lower-7 scores are from those with only 1 or 2 layers. This is interesting since the majority (7/10) of the target labels already exists in source task labels, by which it is reasonable to assume that the necessary information can be provided only with the last layer for those labels. Even in such a situation, however, low-level features contribute to improving the genre classification performance[4].

Among the classes of target task, *classical* and *disco*, *reggae* do not exist in the source task classes. Based on this, I consider two hypotheses, *i)* the performances of those three classes may be lower than the others, *ii)* low-level features may play an important role to classify them since high-level feature from the last layer may be biased to the other 7 classes which exist in the source task. However, both hypotheses are rebutted by

[4]On the contrary, in Task 5 - music emotion classification, high-level feature plays a dominant role (see Section 6.4.2.4).

comparing the performances for each genres with convnet feature '5' and '12345' as in Figure 6.5. First, with '5' convnet feature, *classical* shows the highest accuracy while both *disco* and *reggae* show accuracies around the average accuracy reported over the classes. Second, aggregating early-layer features affects all the classes rather than the three omitted classes. This suggests that the convnet features are not strongly biased towards the genres that are included in the source task and can be used generally for target tasks with music different from those genres.

6.4.2.3 Task 3. Gtzan speech/music classification

Figure 6.6 shows the accuracies of convnet features, baseline feature, and state-of-the-art [158] with low-level features including MFCCs and sparse dictionary learning for Gtzan music/speech classification. A majority of the convnet feature combinations achieve 100% accuracy. MFCC features achieve 99.2%, but the error rate is trivial (0.8% is one sample out of 128 excerpts).

Although the source task is only about music tags, the pre-trained feature in any layer easily solved the task, suggesting that the nature of music and speech signals in the dataset is highly distinctive.

6.4.2.4 Task 4. Music emotion prediction

Figure 6.7 shows the results for music emotion prediction (Task 4). The best performing convnet features achieve 0.633 and 0.415 r^2 scores on arousal and valence axes respectively.

On the other hand, the state-of-the-art algorithm reports 0.704 and 0.500 r^2 scores using music features with a recurrent neural network as a classifier [159] that uses 4,777 audio features including many functionals (such as *quantiles, standard deviation, mean, inter peak distances*) of *12 chroma features*, *loudness*, *RMS Energy*, *zero crossing rate*,

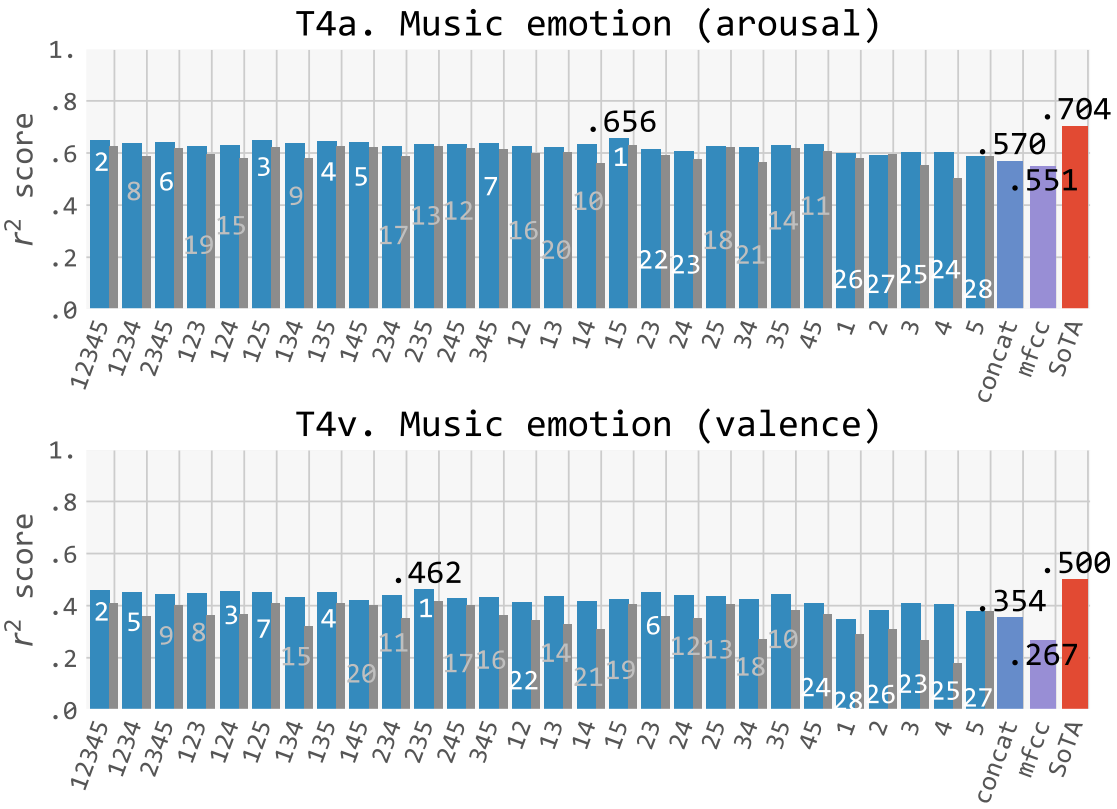

Figure 6.7: Performances of Task 4a (arousal) and 4v (valence) - Music emotion prediction of convnet features (with random convnet features in grey), MFCCs, and the reported state-of-the-art method.

14 MFCCs, spectral energy, spectral roll-off, etc.

For the prediction of arousal, there is a strong dependency on the last layer feature. All top-7 performances are from the feature vectors that include the fifth layer. The first layer feature also seems important, since all of the top-5 strategies include the first and fifth layer features. For valence prediction, the third layer feature seems to be the most important one. The third layer is included in all of the top-6 strategies. Moreover, '3' strategy was found to be best performing among strategies with single layer feature.

To summarise the results, the predictions of arousal and valence rely on different layers, for which they should be optimised separately. In order to remove the effect of the choice of a classifier and assess solely the effect of features, I compare the proposed approach to the baseline method of [159] which is based on the same 4,777 features with

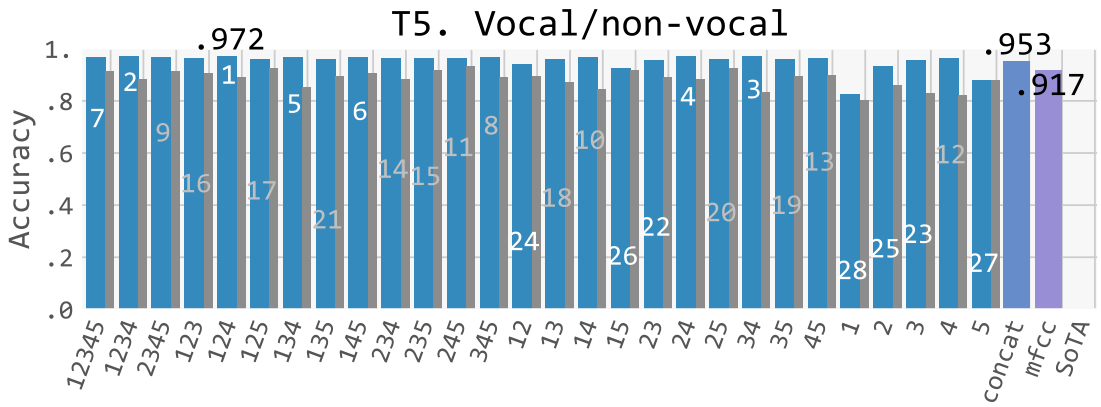

Figure 6.8: Performances of Task 5 - Vocal detection of convnet features (with random convnet features in grey) and MFCCs.

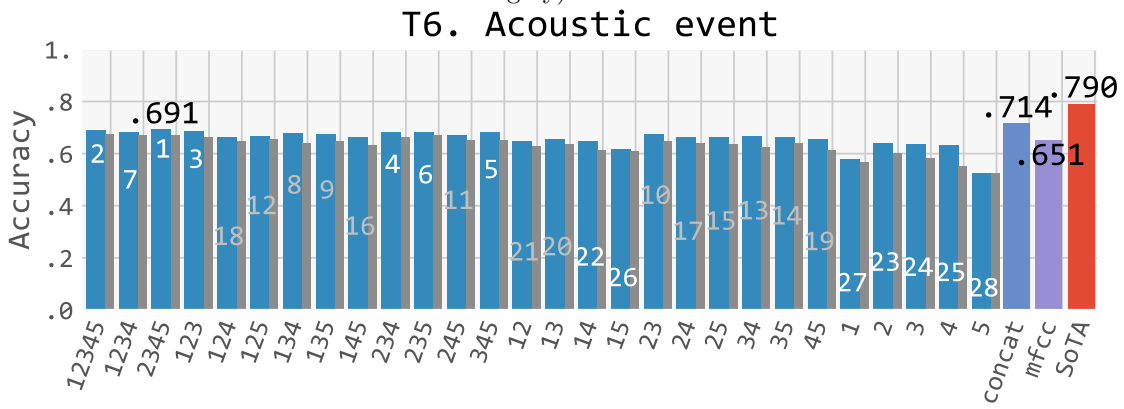

Figure 6.9: Performances of Task 6 - Acoustic event detection of convnet features (with random convnet features in grey), MFCCs, and the reported state-of-the-art method.

SVM, not a recurrent neural network. The baseline method achieves .541 and .320 r^2 scores respectively on arousal and valence, both of which are lower than those achieved by using the proposed convnet feature. This further confirms the effectiveness of the proposed convnet features.

6.4.2.5 Task 5. Vocal/non-vocal classification

Figure 6.8 presents the performances on vocal/non-vocal classification using the Jamendo dataset [143]. There is no known state-of-the-art result, as the dataset is usually used for *frame-based* vocal *detection/segmentation*. Pre-segmented *Excerpt classification* is the task I formulate in this chapter. For this dataset, the fourth layer plays the most

important role. All the 14 combinations that include the fourth layer outperformed the other 14 strategies without the fourth layer.

6.4.2.6 Task 6. Acoustic event detection

Figure 6.9 shows the results on acoustic event classification using Urbansound8K dataset [144]. Since this is not a music-related task, there are no common tags between the source and target tasks, and therefore the final-layer feature is not expected to be useful for the target task.

The strategy of concatenating '12345' convnet features and MFCCs yields the best performance. Among convnet features, '2345', '12345', '123', and '234' achieve good accuracies. In contrast, those with only one or two layers do not perform well. I was not able to observe any particular dependency on a certain layer.

Since the convnet features are trained on music, they do not outperform a dedicated convnet trained for the target task. The state-of-the-art method is based on a deep convolutional neural network with data augmentation [160]. Without augmenting the training data, the accuracy of convnet in the same work is reported to be 74%, which is still higher than my best result (71.4%).[5]

The convnet feature still shows better results than conventional audio features, demonstrating its versatility even for non-musical tasks. The method in [144] with {*minimum, maximum, median, mean, variance, skewness, kurtosis*} of 25 MFCCs and {*mean and variance*} of the first and second MFCC derivatives (225-dimensional feature) achieved only 68% accuracy using the SVM classifier. This is worse than the performance of the best performing convnet feature.

It is notable again that unlike in the other tasks, concatenating convnet feature and MFCCs results in an improvement over either a convnet feature or MFCCs (71.4%).

[5]Transfer learning targeting audio event classification was recently introduced in [72] and achieved a state-of-the-art performance.

This suggests that they are complementary to each other in this task.

6.5 Summary

In this chapter, I proposed a transfer learning approach using deep learning and evaluated it on six music information retrieval and audio-related tasks. The pre-trained convnet was first trained to predict music tags and then aggregated features from the layers were transferred to solve genre classification, vocal/non-vocal classification, emotion prediction, speech/music classification, and acoustic event classification problems. Unlike the common approach in transfer learning, I proposed to use the features from every convolutional layer after applying an average-pooling to reduce their feature map sizes.

In the experiments, the pre-trained convnet feature showed good performance overall. It outperformed the baseline MFCC feature for all the six tasks, a feature that is very popular in music information retrieval tasks because it gives reasonable baseline performance in many tasks. It also outperformed the random-weights convnet features for all the six tasks, demonstrating the improvement by pre-training on a source task. Somewhat surprisingly, the performance of the convnet feature is also very competitive with state-of-the-art methods designed specifically for each task. The most important layer turns out to differ from task to task, but concatenating features from all the layers generally worked well. For all the five *music* tasks, concatenating MFCC feature onto convnet features did not improve the performance, indicating the music information in MFCC feature is already included in the convnet feature.

Earlier in the chapter, I mentioned there is a hierarchy among the trained convolution kernels in a deep convnet. This is a well-known property that is revealed by some interesting methods to look into a trained network. In the following chapter, I introduce my extension of one of those methods to music signal.

Chapter 7

Understanding Convnets for Music Classification

Abstract

In this chapter, which is the last chapter of the main body of this thesis, I introduce research about interpreting the neural networks by analysing some trained networks. Although each experiment in this chapter is based on a specifically trained network, I focus on the properties that would generalise in (convolutional) neural networks for music classification including tagging.

By understanding a trained deep neural network, we can expand our knowledge of how it works, what would be its potential problem, and even the domain, in our case, music and tags. I propose two methods for this. First, auralisation of trained features shows what are learned when the network learn to classify music items. It shows that the early layers learn the low-level, onset-related features while the deep layers represent some time-frequency patters, or sound textures, which construct a part of timbre. Second, label vector analysis showed how a trained network sees the relationships of music tags, revealing some unexpected similarity between tags.

The works introduced in this chapter are based on my papers, "Auralisation of Deep Convolutional Neural Networks: Listening to Learned Features" [13] and "The Effects of Noisy Labels on the Deep Convolutional Neural Networks for Music Classification" [7].

The auralisation was done with music genre classification, not music tagging, because it was done during my research internship in 2015 with a private genre classification dataset of the company and the experiment involves listening many clips to interpret the learned features.

7.1 Introduction

Deep neural networks are often accused of being 'black box' although anyone can investigate inside the 'box' which simply consists of numbers. However, it is not a totally

false accusation because the problem still exists when there are easily *millions* of boxes.

The first method is the one that is named 'auralisation'. As the name indicates, it is a method to understand a trained convnet by auralising (or sonifying) the trained kernels of convolutional neural networks. This is an extension of a popular work that visualises the trained kernels [131] and helps to understand convnets that are trained for music classification tasks.

The second method utilises 'label vectors' which represents how the networks combine the trained feature to make the predictions in the output layer. This approach directly interprets the behaviour of a trained network with respect to the output labels.

7.2 Auralisation

7.2.1 Motivation

The mechanism of feature learning in convnets is relatively clear when the target shapes are known. For example, the convolution filters that are learned for chord recognition [57] or transcription [58] are not mystified; they are trained to capture certain spectral distributions of interest. What has not been demystified yet is how convnets work for tasks such as mood recognition or genre classification. Those tasks are related to subjective, perceived impressions, which are yet in questions. Their relation to acoustical properties and whether neural networks models can learn a relevant and optimal representation of sound that helps to boost performance in these tasks is an open question. As a result, researchers currently lack in understanding of what is learned by convnets when convnets are used in those tasks, even if it shows state-of-the-art performance [4, 54].

One effective way to examine convnets was introduced in [131], where the features in deeper levels are visualised by a method called *deconvolution*. Deconvolving and unpooling layers enable people to see which part of the input image are focused on by

each filter. However, it does not provide a complete, relevant explanation of convnets on music. Unlike computer vision, where outlines of images play an important role in many tasks, spectrograms mainly consist of continuous, smooth gradients. There are not only local correlations but also global correlations such as harmonic structures and rhythmic events. These differences inspire to develop a new approach to understand convnets for music.

7.2.2 Visualisation of convnets

The behaviour of a convnet is not deterministic as the operation of max-pooling varies by the input. This is why analysing the learned weights of convolutional layers does not provide satisfying explanations.

Instead, a way to understand a convnet is to visualise the features given different inputs. Visualisation of convnets was introduced in [131], which showed how high-level features (postures/objects) are constructed by combining low-level features (lines/curves), as illustrated in Figure 7.1. In the figure, the shapes that features represent evolve. In the first layers, each feature simply responds to lines with different directions. By combining them, the features in the second and third layers can capture certain shapes - a circle, textures, honeycombs, etc. During this forward path, the features not only become more complex but also allow slight variances, and that is how the similar but different faces of dogs can be recognised by the same feature in Layer 5 in Figure 7.1. Finally, the features in the final layer successfully capture the outlines of the target objects such as cars, dogs, and human faces.

7.2.3 Auralisation of convnets

Although we can obtain spectrograms by deconvolution, deconvolved spectrograms do not necessarily facilitate an intuitive explanation. This is because seeing a spectrogram does not necessarily provide clear intuition that is comparable to observing an image.

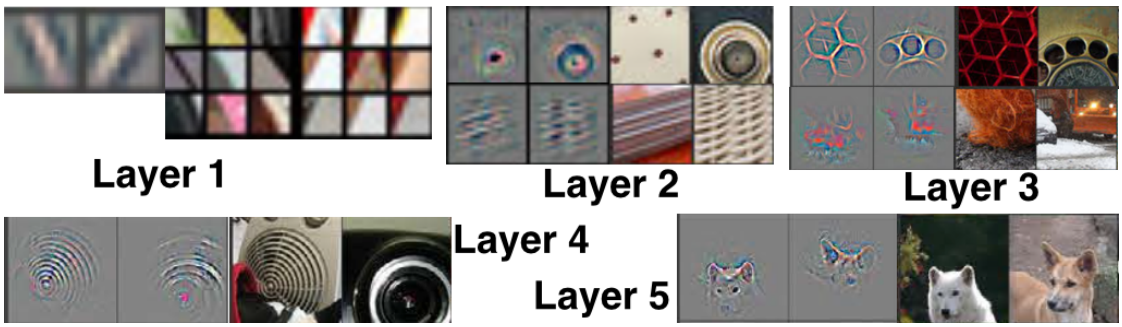

Figure 7.1: Deconvolution results of convnets trained for image classification. In each layer, the responses of the filters are shown on the left with gray backgrounds and the corresponding parts from images are shown on the right. Image is courtesy of [131].

To solve this problem, I propose to reconstruct audio signals from deconvolved spectrograms, which is called *auralisation*. This requires an additional stage for inverse-transformation of a deconvolved spectrogram. The phase information is provided by the phase of the original time-frequency representations, following the generic approach in spectrogram-based sound source separation algorithms. STFT is therefore recommended as it allows us to obtain a time-domain signal easily.

A pseudo code of the auralisation is described in Listing 7.2.3. Line 1 indicates that we have a convolutional neural network that is trained for a target task. In line 2-4, an STFT representation of a music signal is provided. Line 5 computes the weights of the neural networks with the input STFT representation and the result is used during the deconvolution of the filters in line 6 ([131] for more details). Line 7-9 shows that the deconvolved filters can be converted into time-domain signals by applying the phase information and inverse STFT.

```
convnet_model  = train_convnets(*args)  # convnet model
src = load(wavfile)
SRC = stft(src)
weights = unpool_info(convnet_model, SRC.mag)
deconved_imgs = deconv(weights, SRC.mag)  # deconvolution
for img in deconved_imgs:
    signal = inverse_stft(img * SRC.phase)  # apply inverse STFT
    wav_write(signal)  # output the auralised signal
```

Listing 7.1: A pseudo-code of auralisation procedure

7.2.4 Experiments and discussions

I implemented a convnet for music genre classification with a dataset obtained from *Naver Music* [1]. All audio signal processing was done using *Librosa* [161]. Three genres (*ballad*, *dance*, and *hip-hop*) were classified using 8,000 songs in total. In order to maximally exploit the data, 10 clips of 4 seconds were extracted for each song, generating 80,000 data samples by STFT. STFT is computed with 512-point windowed Fast Fourier Transform with 50% hop size and sampling rate of 11,025 Hz. 6,600/700/700 songs were designated as training/validation/test sets respectively.

The convnet architecture consists of 5 convolutional layers of 64 feature maps and 3-by-3 convolution kernels, max-pooling with size and stride of (2,2), and two fully connected layers as illustrated in the Figure 7.2. It is similar to Compact-convnet. This system showed 75% of the classification accuracy on average.

It is noteworthy that the homogeneous size of convolutional kernels (3×3) results in the different effective widths and heights by layer. This is due to the subsampling and summarised in the table 7-A.

[1] http://music.naver.com, a Korean music streaming service

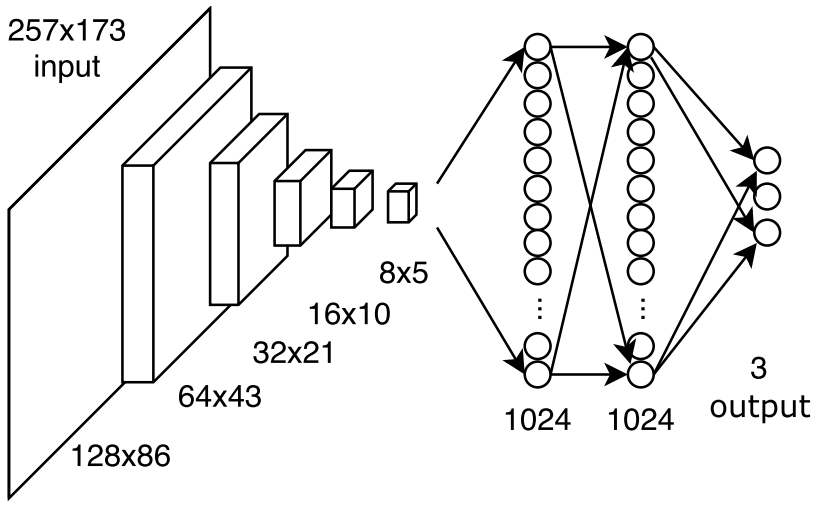

Figure 7.2: A block diagram of the trained convnet for genre classification.

Layer	Convolution	Effective width	Effective height
1	3×3	93 ms	86 Hz
2	3×3	162 ms	151 Hz
3	3×3	302 ms	280 Hz
4	3×3	580 ms	538 Hz
5	3×3	1137 ms	1270 Hz

Table 7-A: The effective sizes of convolutional kernels

7.2.4.1 Deconvolved results with music excerpts

The deconvolution-based approach is example-based. In this chapter, I selected four music signals; a piano solo piece by *Bach*, a Korean popular song with a female vocalist by *Lena Park (Dream)*, another Korean popular song with a male vocal by *Toy*, and a rap song by *Eminem*. Table 7-B provides more information about the music items. In the following section, several selected examples of deconvolved spectrograms are illustrated with descriptions.[2] In each figure, the selected features are presented with a musical interpretation. During the overall process, listening to auralised signals helped to identify patterns in the learned features.

[2]The results are demonstrated on-line at `http://wp.me/p5CDoD-k9`. An example code of the deconvolution procedure is released at `https://github.com/keunwoochoi/Auralisation`

Name	Summary
Bach	Classical, piano solo
Dream	Pop, female vocal, piano, bass guitar
Toy	Pop, male vocal, drums, piano, bass guitar
Eminem	Hip-hop, male vocal, piano, drums, bass guitar

Table 7-B: Descriptions of the four selected music items

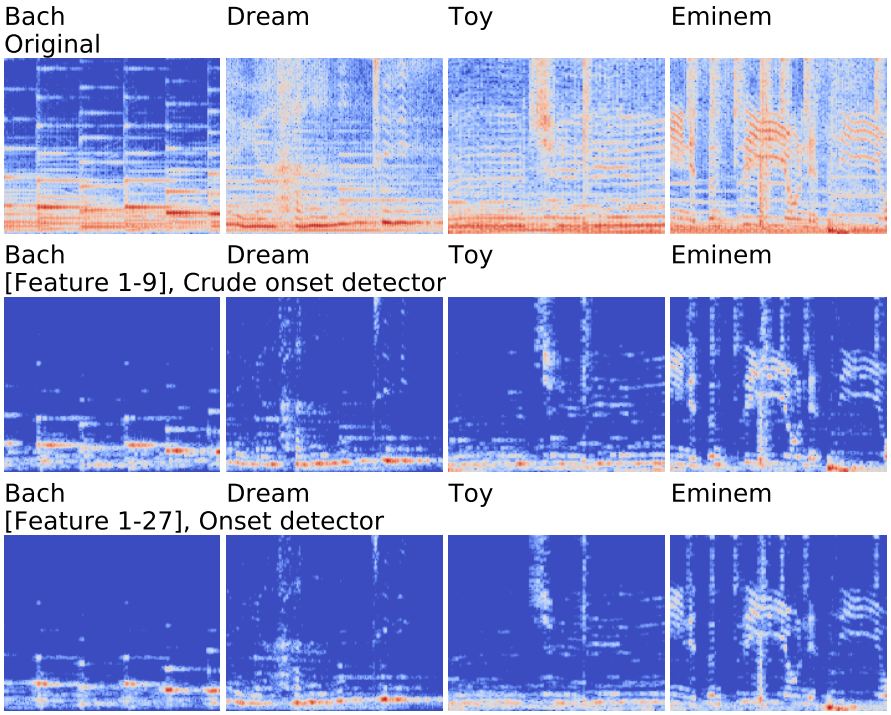

Figure 7.3: Spectrograms of deconvolved signal in Layer 1

7.2.4.2 Layer 1

(a) Feature 1-9 (b) Feature 1-27

Figure 7.4: The learned weights of Features (a) 1-9 and (b) 1-27. The distributions along rows and columns roughly indicate high-pass filter behaviours along x-axis (time axis) and low-pass filter behaviours along y-axis (frequency axis). As a result, they behave as onset detectors.

In Layer 1, I selected two features to present their deconvolved spectrograms as well

as the corresponding weights. Because the weights in the first layer are applied to the input directly without max-pooling, the mechanism are determined and can be analysed by inspecting the weights regardless of input. For instance, Feature 1-9 (9th feature in Layer 1) and Feature 1-27, works as an onset detector. The learned weights are shown in Figure 7.4. By inspecting the numbers, it can be easily understood that the network learned to capture vertical lines. In spectrograms, vertical lines often correspond to the time-frequency signature of percussive instruments.

The roles of learned features are not mutually exclusive in Layer 1. For example, many features in Layer 1 turn out to learn to represent onset detectors or suppressors. It is a similar result to the result that is often obtained in visual image recognition, where convnets learn line detectors with various directions (also known as *edge detectors*). With spectrograms, however, the network focuses on detecting horizontal and vertical edges rather than diagonal edges. This may be explained by the energy distribution in spectrograms. Horizontal and vertical lines are main components of harmonic and percussive instruments, respectively, while diagonal lines mainly appear in the case of frequency modulation, which is probably less common.

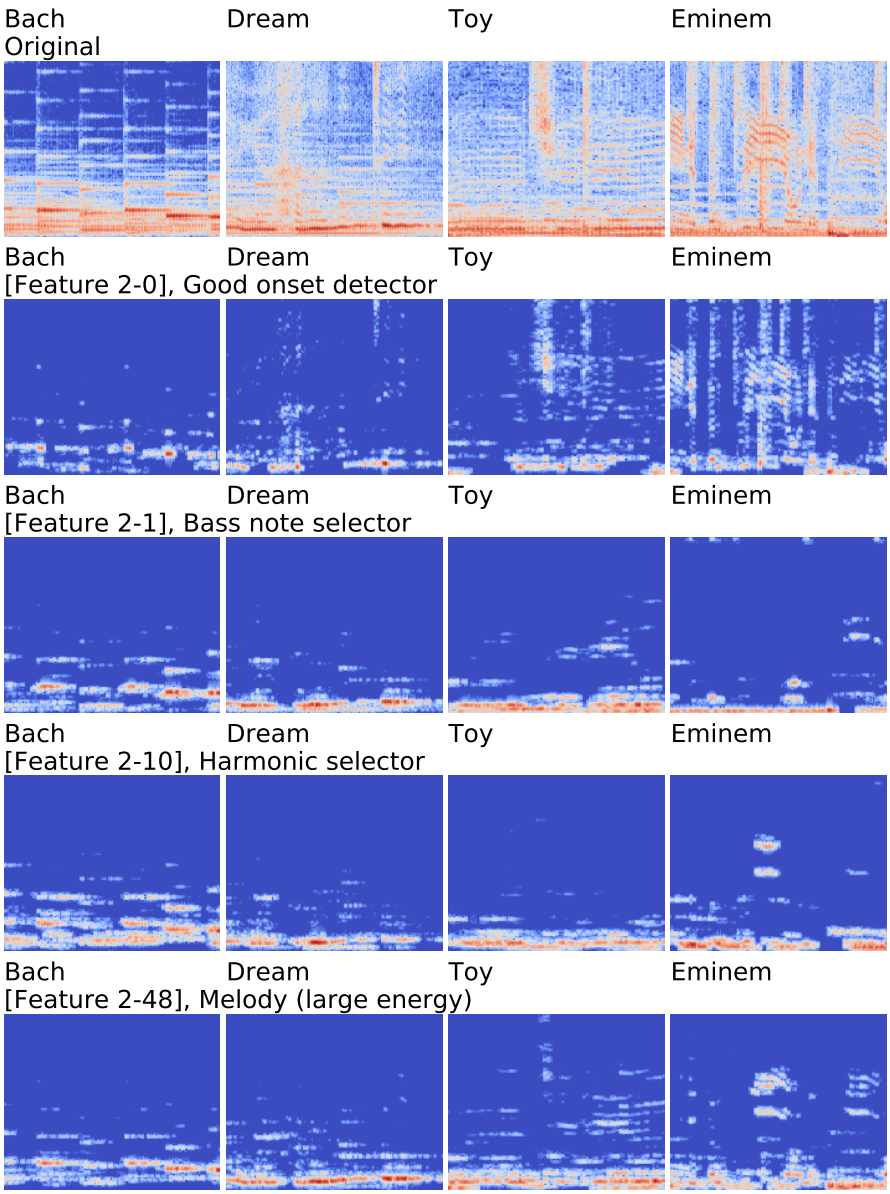

Figure 7.5: Spectrograms of deconvolved signal in Layer 2

7.2.4.3 Layer 2

Layer 2 shows more evolved, complex features compared to Layer 1. Feature 2-0 is an advanced (or stricter) onset detectors than the onset detectors in Layer 1. This improvement can be explained from two perspectives, which are not mutually exclusive. First, as the features in Layer 2 can cover a wider range both in time and frequency axis

than in layer 1, non-onset parts of signals can be suppressed more effectively. Second, the multiple onset detectors in Layer 1 can be combined, enhancing their effects.

Feature 2-1 (*bass note*), roughly selects the lowest fundamental frequencies given harmonic patterns. Feature 2-10 behaves as a harmonic component selector, excluding the onset parts of the notes. Feature 2-48 is another harmonic component selector with small differences. It behaves as a short melodic fragments extractor, presumably by extracting the most salient harmonic components.

7.2.4.4 Layer 3

The patterns of some features in Layer 3 are similar to that of Layer 2. However, some of the features in Layer 3 contain higher-level information e.g. focusing on different instrument classes.

The deconvolved signal from Feature 3-1 consists of onsets of harmonic instruments, being activated by voices and piano sounds but not highly activated by hi-hats and snares. The sustain and release parts of the envelopes are effectively filtered out in this feature. Feature 3-7 is similar to Feature 2-48 but it seems more accurate at selecting the fundamental frequencies of top notes. Feature 3-38 extracts the sounds of kick drum with a very good audio quality. Feature 3-40 effectively suppresses transient parts, resulting in softened signals.

The learned features imply that the roles of some learned features are analogous to tasks such as harmonic-percussive separation, onset detection, and melody estimation.

7.2.4.5 Layer 4

Layer 4 is the second last layer of the convolutional layers in the architecture and expected to represent high-level features. In this layer, a kernel covers a large area (580 ms$\times$538 Hz) and the size affects the deconvolved spectrograms. The activation is rather sparse

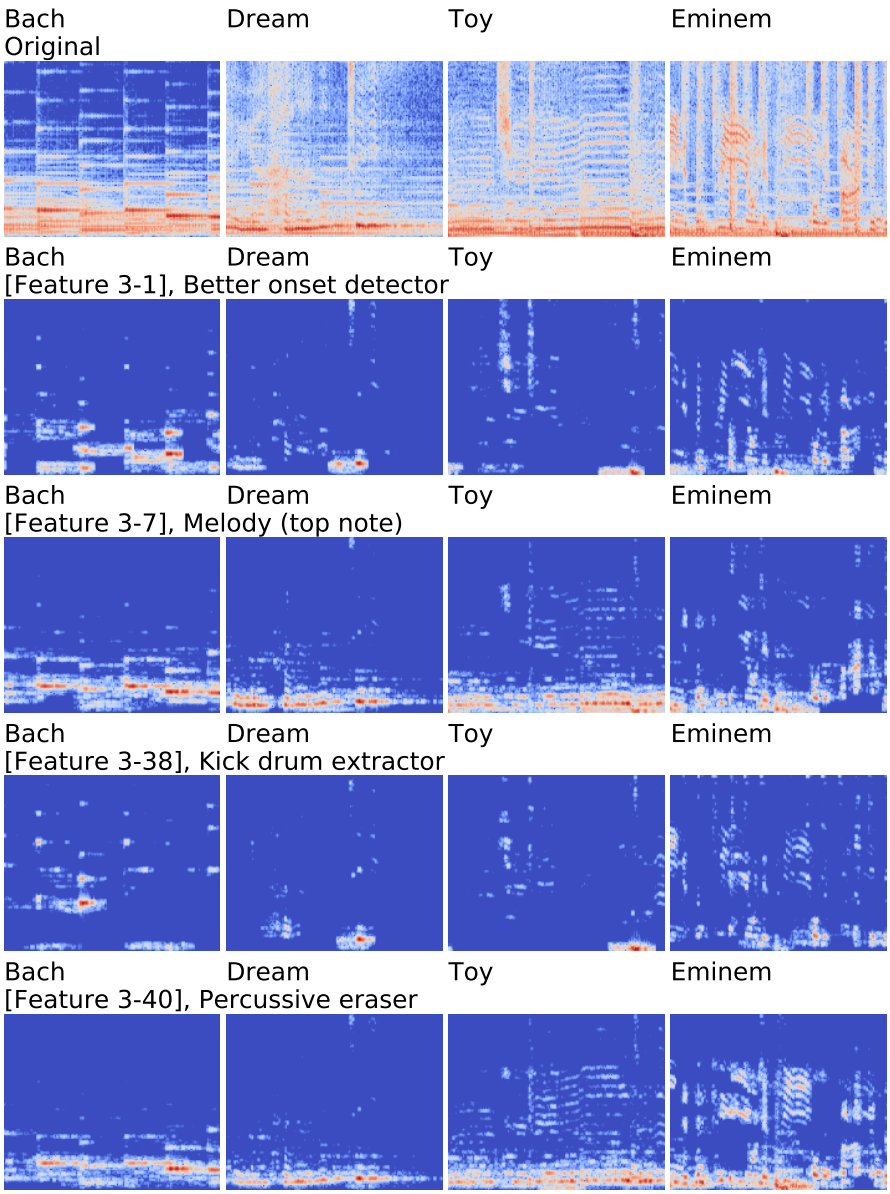

Figure 7.6: Spectrograms of deconvolved signal in Layer 3

due to combined ReLU activations are applied to a larger area. It becomes trickier to name the features by their characteristics, although their activation behaviours are consistent. Feature 4-11 removes vertical lines and captures another harmonic texture. Although the coverages of the kernels increase, the features in Layer 4 try to find patterns more precisely rather than simply respond to common shapes such as edges as done in the early layers. As a result, the activations become more sparse because a feature

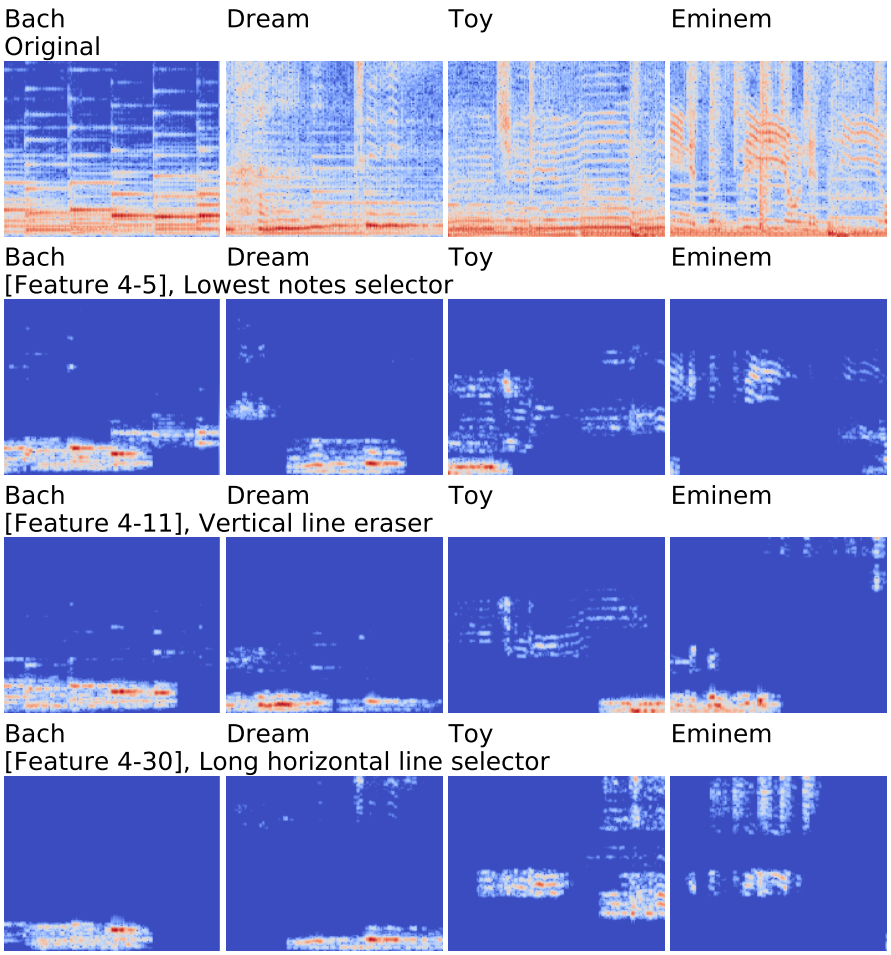

Figure 7.7: Spectrograms of deconvolved signal in Layer 4

responds only if a certain, unusual pattern.

7.2.4.6 Layer 5

This is the final convolutional layer of the convnet and therefore it represents the highest-level features among all the learned features. At this stage, naming the feature is tricky by either listening to auralised signals or seeing deconvolved spectrograms. Feature 5-11, 5-15, and 5-33 are therefore named as textures. Feature 5-56, harmo-rhythmic texture, is activated almost only if strong percussive instruments *and* harmonic patterns overlap. Feature 5-33 is completely inactivated with the fourth excerpt, which is the only Hip-Hop

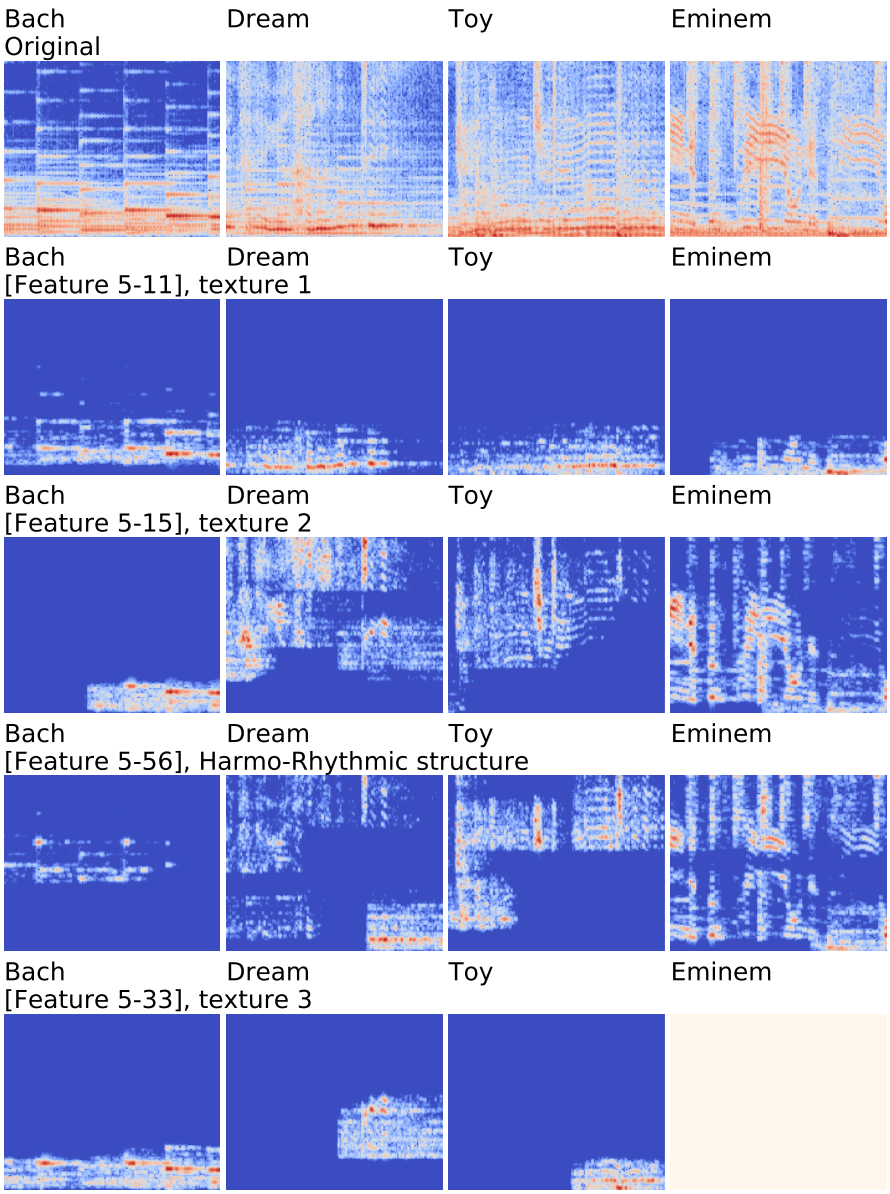

Figure 7.8: Spectrograms of deconvolved signal in Layer 5

music, suggesting it may be useful for classification of (non-)Hip-Hop.

7.2.5 Limitations

The proposed approach have some limitations, which is summarised as below.

- Auralisation requires listening, therefore it has some limitations that are common in audio listening tests. Listening is time-consuming and subjective.
- Although the motivation was *to listen* instead of *to see*, sometimes the characteristics of the deconvolved item are subtle and easier to find by visual inspection than aural inspection.
- The auralising process itself is scalable but listening the result is not. This problem may be address by pruning less-important kernels before auralisation so that the number of items to listen to can be reduced.

7.3 Label vector analysis

7.3.1 Introduction

In the previous sections, I investigated the learned features by feeding the network with spectrograms, focussing on the input side of the network. In this section, I present label vector analysis, which is more about the network and the output label.

It is worth considering how the groundtruth is distilled into the network after training, and whether we can leverage the trained network beyond our particular task. To answer these questions we use the trained network weights to assess how the network 'understands' music content by its label. This analysis also provides a way to discover unidentified relationships between labels and the music. The goal of label vector analysis is to better understand network training as well as assess its capacity to represent domain knowledge, i.e., relationships between music tags that is not explicitly shown in the data.

7.3.2 Compact-convnet and LVS

I use the Compact-convnet that introduced in Section 2.4 and described again in Table 2-B. To remind the details, it was trained for predicting 50 music tags. In Compact-convnet the output layer has a dense connection to the last convolutional layer. The weights are represented as a matrix $\mathbf{W} \in \mathbb{R}^{N \times 50}$, where N is the number of feature maps (N=32 for our case) and 50 is the number of the predicted labels.

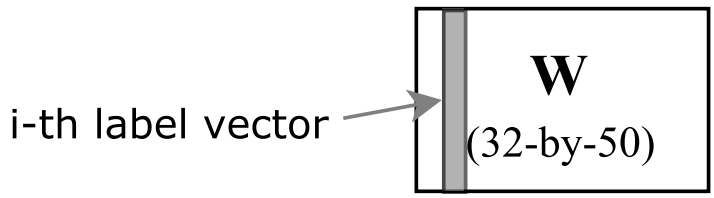

Figure 7.9: Label vector

What really matters in this chapter is the method to analyse the network after training. In the prediction phase, the columns of $\mathbf{W}$ can be interpreted as N-dimensional latent vectors since they represent how the networks combine information in the last convolutional layer to make the final prediction. We call these *label vector*s.

The pairwise label vector similarity (LVS) is computed using the dot product, i.e., $S(i,j) = w(i) \cdot w(j)$ where $i, j \leq 50$ or equivalently:

$$\mathbf{S} = \mathbf{W}^\top \cdot \mathbf{W}, \tag{7.1}$$

which yields a 50×50 symmetric matrix.

7.3.3 Results

LVS is illustrated in Figure 7.10. The pattern is similar to the values in NCO (normalised co-occurrence, denoted as $C(i,j)$) shown in Figure 7.11 (see Chapter 3) which shows the co-occurrence pattern in the groundtruth that the network is trained against. On average, the similarities in $S(i,j)$ are higher than those in $C(i,j)$. In $\mathbf{S}$, only four pairs show negative values, 'classic rock' – 'female vocalists', and 'mellow' – {'heavy metal', 'hard rock', and 'dance'}. In other words, label vectors are distributed in a limited space corresponding to a 32 dimensional vector space, where the angle θ between $w(i)$ and $w(j)$ is smaller than $\pi/2$ for most of the label vector pairs. This result can be interpreted in two ways: how well the convnet reproduce the co-occurrence that was provided by the training set; and if there is additional insight about music tags in the trained network.

First, the Pearson correlation coefficient of the rankings by LVS and NCO is 0.580.[3]. The top 20 most similar label pairs are sorted and described in Table 7-C. The second row of the table shows similar pairs according to the label vectors estimated by the network. Eleven out of 20 pairs overlap with the top 20 NCO tuples shown in the top row of the table. Most of these relations can be explained by considering the genre hierarchy.

[3]Because of the asymmetry of $C(i,j)$, rankings of $\max(C(i,j), C(y,y))$ are used.

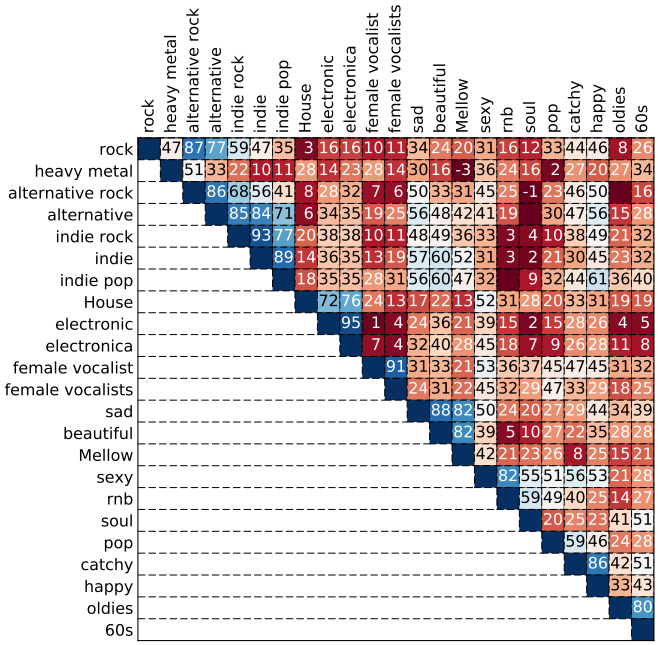

Figure 7.10: Label vector similarity matrix by Eq. 7.1 (of manually selected 23 tags, same in Figure 7.11, where symmetric components are omitted and numbers are $\times 100$ after dot product for visual clarity.

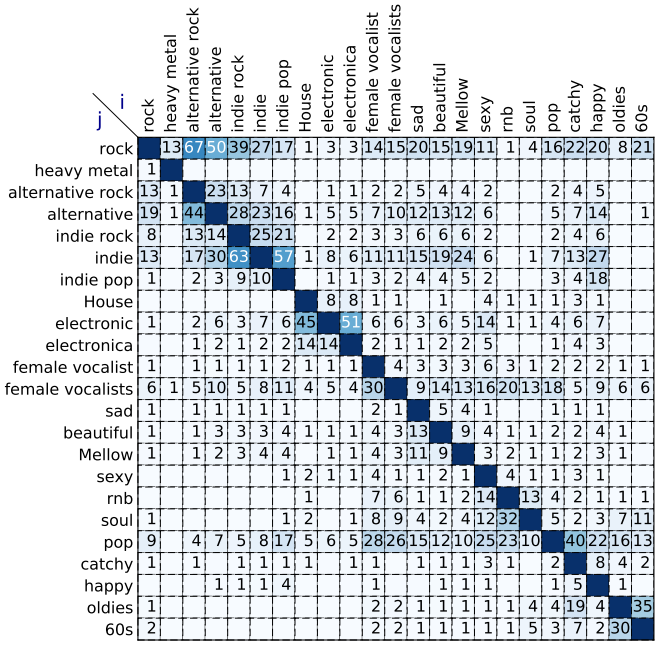

Figure 7.11: Normalised tag co-occurrence pattern of the selected 23 tags from the training data. For the sake of visualisation, we selected 23 tags out of 50 that have high co-occurrences and represent different categories; genres, instruments and moods. The values are computed using Eq. 3.1 (and are multiplied by 100, i.e., shown in percentage), where y_i and y_j respectively indicate the labels on the x-axis and y-axis. (This is an identical figure to Figure)

Table 7-C: Top-20 Similar tag tuples by two analysis approaches. The first row is by analysing co-occurrence of tags in groundtruth. The second row is by the similarity of trained label vector (see 7 for details). Common tuples are annotated with matching symbols.

Similar tags by groundtruth labels	(alternative rock, rock)§ (indie rock, indie)$^{\#}$ (House, dance)‡‡ (indie pop, indie) (classic rock, rock) (electronica, electronic)* (alternative, rock) (hard rock, rock) (electro, electronic)** (House, electronic) (alternative rock, alternative)¶ (catchy, pop) (indie rock, rock) (60s, oldies)†† (heavy metal, metal)§§ (rnb, soul) (ambient, electronic) (90s, rock) (heavy metal, hard rock)‡ (alternative, indie)$^{\|}$
Similar tags by label vectors	(electronica, electronic)* (indie rock, indie)$^{\#}$ (female vocalist, female vocalists) (heavy metal, hard rock)‡ (indie, indie pop) (sad, beautiful) (alternative rock, rock)§ (alternative rock, alternative)¶ (happy, catchy) (indie rock, alternative) (alternative, indie)$^{\|}$ (rnb, sexy) (electro, electronic)** (sad, Mellow) (Mellow, beautiful) (60s, oldies)†† (House, dance)‡‡ (heavy metal, metal)§§ (chillout, chill) (electro, electronica)

Besides, pairs such as ('female vocalists', 'female vocalists') and ('chillout', 'chill') correspond to semantically similar words. Overall, tag pairs showing high similarity (LVS) reasonably represent musical knowledge and correspond to high NCO values computed from the ground truth. This confirms the effectiveness of the network to predict subjective and high-level semantic descriptors from audio only – the LVS, which is computed from audio-based feature, approximated the NCO.

Second, there are several pairs that are *i)* high in LVS, *ii)* low in NCO, and *iii)* presumably music listeners would reasonably agree with their high similarity. These pairs show the extracted representations of the trained network can be used to measure tag-level musical similarities even if they are not explicitly shown in the groundtruth. For example, pairs such as ('sad', 'beautiful'), ('happy', 'catchy') and ('rnb', 'sexy') are in the top 20 of LVS (6^{th}, 9^{th}, and 12^{th} similarities with 0.88, 0.86, and 0.82 of similarity values respectively). On the contrary, according to the ground truth, they are only 129^{th}, 232^{nd}, 111^{th} co-occurring with 0.13, 0.08, and 0.14 of co-occurrence likelihood respectively.

7.4 Summary

In this chapter, I introduced two methods and experiments results, both of which aim to explain the trained convnets.

The first approach, auralisation, is an example-based investigation of a convnet that is extended to music signals. This is done by inverse-transformation of deconvolved spectrograms to obtain time-domain audio signals. Listening to the audio signal enables researchers to understand the mechanism of convnets that are trained with audio signals. In the experiments, I trained a 5-layer convnet to classify genres. Selected learnt features are reported with interpretations from musical and music information aspects.

The second approach, label vector analysis, is a more direct investigation of a trained convnet regardless of the input signals. The analysis based on LVS indirectly validates that the network learned meaningful representations that correspond to the groundtruth. Moreover, we found several pairs that are considered similar by the network which may help to extend our understanding of the relation between music and tags.

Both of the approaches help us to understand the 'black boxes' of deep neural networks when they are trained for music classification. The auralisation results showed the different levels of abstraction in the layers of the deep convnet which is trained for music genre classification. The label vector analysis showed the convnet reproduce the provided groundtruth and the training even yield a network that can analyse the musical content that are not explicitly given by the training data.

Chapter 8

Conclusion

8.1 Summary

I have presented various topics that are related to music tagging and deep neural networks. Through chapters 3 to 7, I have proposed motivations, methods, and experiments and showed results that contribute to *understanding*, *utilising*, *comparing*, and *re-using* convnet-based music taggers.

In Chapter 3, I quantified the noise on the labels of the largest public music tagging dataset, the million song dataset, as well as its effect on the training and evaluation of convnet-based music taggers. I could observe considerable label noise, even over 50% of error rates for certain labels. Based on the observation, I defined *tagability* which *i)* indicates how a tag is likely to be positively labelled under the weakly-labelling scheme and *ii)* is shown to be a measure of the performance degradation per tag. I observed that label noise was negatively correlated with tagability which is positively correlated to the classification performance. I could draw a conclusion that low unusualness (which is positively correlated to tagability) results in the noisy label, which then harms the training.

In Chapter 4, I proposed a convolutional neural network structure for music tagging. Its performance was extensively compared with the performance of other structures. The comparison was done by controlling the number of parameters. With the results, the performances were presented against *i)* the number of parameters ($\sim$ the memory usage) and *ii)* training time ($\sim$ the computation complexity). In the experiment, 2D convnet and convolutional recurrent neural networks achieved the best performances with fixed sizes of parameters, showing a trade-off between computation time and memory usage. The results showed that recurrent layers can be used for better performance while a simpler, fully convolutional network may be preferred for a quicker training and prediction.

In Chapter 5, I showed the comparison results of convnet-based music tagger with varying the audio signal processing methods. The experiment result showed a logarithmic mapping of the time-frequency magnitude is crucial while the network turned out to be resilient to other distortions.

In Chapter 6, I proposed a transfer learning method based on convnet music tagger. Transfer learning is to use features from a trained deep neural network to a problem with a smaller dataset (which is the case of many MIR problems) and/or for those who cannot utilise a hardware resource for training deep neural networks. Unlike usual transfer learning approaches, I proposed to use the feature maps of all layers. The proposed feature resulted in strong performances in the five selected MIR tasks, outperforming MFCCs and many audio descriptors.

In Chapter 7, I presented two methods to investigate and explain convnets for music classification. The first method is auralisation, which consists of converting the learned features into audio signals so that we can attempt to understand them by listening. The second method is label vector analysis, which showed the relationship between the classes from the perspective of the trained neural network. With auralisation results I showed that there are some low-level, local features learned in the earlier layers, some of which are related to onsets; while there are high-level features learned in the following

layers, overall, constituting a feature hierarchy, which would be worth analysing deeper. With label vector analysis, I could find that the neural networks can deal with synonym problem well although the label did not necessarily co-occur. This is done by the network being able to learn some representations that are linked to the audio content that contributes to the label.

8.2 Conclusion

Let's recall the fundamental hypothesis presented in Chapter 1.

We find effective deep learning practices for the task of music tagging that improves the classification performance

I have been providing a good evidence to show that with the appropriate architecture, the appropriate input preprocessing, understanding of the dataset, and understanding of neural networks, we can automatically tag music audio signals in a meaningful and useful way.

By appropriate architecture, the convolutional recurrent neural network architecture introduced in Chapter 4 achieved a better performance than existing architectures.

By appropriate input preprocessing, the logarithmic compression of time-frequency magnitude turns out to be effective while other preprocessing methods did not result in affecting the performance as in Chapter 5.

By understanding of the dataset, the label noise and its effects turn out to depend on human annotator's subjective bias on whether labels are worth to tag, which was defined as 'tagability' as in Chapter 3.

By understanding of neural networks, the trained feature hierarchy and their

relationship to the audio input were analysed in Chapter 7. Based on the insights, a novel transfer learning method was proposed to use all of the convolutional layers as in Chapter 6.

Although deep learning is occasionally referred to 'end-to-end' learning, domain knowledge is still required by practitioners to efficiently use the given resources (data, hardware, and time). To elaborate, it is important to let the learning focus on the most training-worthy part, e.g., layers that connects the time-frequency representations to the target label, while the rest (e.g., input data preprocessing and designing network structures) can be decided based on the domain knowledge. In this manner, the knowledge and insights presented in this thesis will help the community to build a successful music classifier based on deep neural networks by providing *new domain knowledge* of music in deep learning. Lastly, however, it is worth noting that in general, the research community is moving towards more 'end-to-end' approaches, e.g., audio sample-based music tagger [77].

8.3 Outlook

8.3.1 On music classification

The experience of using deep neural networks led me to realise the limitations of the current approaches. Among many aspects of music signals, 2-dimensional convnets seem to focus on the 'sound texture', which is merely an aspect of timbre, while unpurposedly ignoring other musical components – temporal envelope, tempo, rhythms, chords, or melodies. My preliminary experiments on music similarity with convnet features resulted in giving me a similar impression and [162] also showed that low-level statistics (the texture) play an important role in visual image recognition. Although exploiting the current approach is providing even

better performance [77], in a long term, the research would explore to take advantage of diverse aspects of music, parting from computer vision-motivated methods. Recently, there have been some attempts at rhythm [76], [163] and multi-modal music classification [164], and I believe there is a huge potential to explore in this direction.

8.3.2 On MIR task formulation

So far, many MIR problems have been naturally proposed and formulated to automate existing music-related tasks. For example, genre classification or music transcription are conventional tasks that were only done manually, and methods have been developed to automate the job. Apparently, this means that solving those tasks are meaningful. However, they require manual annotation which is not scalable. This makes it hard to address them with data-driven approaches or to adopt them as intermediate/source tasks for other MIR problems. Tagging is relatively free from this problem and that is why it has been a popular topic since deep learning became widely used in the field. Moreover, many tasks involve subjective annotation, e.g., mood prediction, which raises a fundamental question of the glass ceiling of the performance.

I expect people would be interested in finding a new solution for a better transfer learning by explicitly relying on large-scale data. For example, instrument recognition is a mid-level MIR task, which may help to solve higher-level tasks such as genre classification. The training samples can be quickly synthesised by mixing two labelled music signals - while they might be musically pleasing or even plausible, they can be used to build a large dataset for a proxy task. Music source separation can be done in a similar fashion, therefore would be one of the candidates. There are open issues such as how the synthesising should be done to make the task meaningful enough and this would be already a big research topic. Still, this can be less

challenging than obtaining order-of-magnitude bigger datasets compared to ones we currently have, or developing algorithms that only requires order-of-magnitude smaller datasets than deep neural networks are currently demanding.

References

[1] Douglas Eck, Paul Lamere, Thierry Bertin-Mahieux, and Stephen Green, "Automatic generation of social tags for music recommendation," in *Advances in neural information processing systems*, 2008, pp. 385–392.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] 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," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[4] Sander Dieleman and Benjamin Schrauwen, "End-to-end learning for music audio," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6964–6968.

[5] Keunwoo Choi, György Fazekas, Kyunghyun Cho, and Mark Sandler, "A tutorial on deep learning for music information retrieval," *arXiv preprint arXiv:1709.04396*, 2017.

[6] Paul Lamere, "Social tagging and music information retrieval," *Journal of new music research*, vol. 37, no. 2, pp. 101–114, 2008.

[7] Keunwoo Choi, György Fazekas, Kyunghyun Cho, and Mark Sandler, "The effects of noisy labels on deep convolutional neural networks for music tagging," February 2018, vol. 2, IEEE.

[8] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho, "Convolutional recurrent neural networks for music classification," in *2017 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2017.

[9] Keunwoo Choi, Deokjin Joo, and Juho Kim, "Kapre: On-gpu audio preprocessing layers for a quick implementation of deep neural network models

with keras," in *Machine Learning for Music Discovery Workshop at 34th International Conference on Machine Learning*. ICML, 2017.

[10] Keunwoo Choi, György Fazekas, Kyunghyun Cho, and Mark Sandler, "A comparison of audio signal preprocessing methods for deep neural networks on music tagging," *arXiv:1709.01922*, 2017.

[11] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho, "Transfer learning for music classification and regression tasks," in *The 18th International Society of Music Information Retrieval (ISMIR) Conference 2017, Suzhou, China*. International Society of Music Information Retrieval, 2017.

[12] Keunwoo Choi, György Fazekas, and Mark Sandler, "Explaining deep convolutional neural networks on music classification," *arXiv preprint arXiv:1607.02444*, 2016.

[13] Keunwoo Choi, Jeonghee Kim, György Fazekas, and Mark Sandler, "Auralisation of deep convolutional neural networks: Listening to learned features," in *International Society of Music Information Retrieval (ISMIR), Late-Breaking/Demo Session, Malaga, Spain*. International Society of Music Information Retrieval, 2015.

[14] Keunwoo Choi, György Fazekas, and Mark Sandler, "Text-based lstm networks for automatic music composition," *arXiv preprint arXiv:1604.05358*, 2016.

[15] Keunwoo Choi, György Fazekas, and Mark Sandler, "Automatic tagging using deep convolutional neural networks," in *The 17th International Society of Music Information Retrieval Conference, New York, USA*. International Society of Music Information Retrieval, 2016.

[16] Pasi Saari, Mathieu Barthet, György Fazekas, Tuomas Eerola, and Mark Sandler, "Semantic models of musical mood: Comparison between crowd-sourced and curated editorial tags," in *Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1–6.

[17] Jongpil Lee and Juhan Nam, "Multi-level and multi-scale feature aggrega-

tion using pre-trained convolutional neural networks for music auto-tagging," *arXiv preprint arXiv:1703.01793*, 2017.

[18] Iván Cantador, Ioannis Konstas, and Joemon M. Jose, "Categorising social tags to improve folksonomy-based recommendations," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 9, no. 1, pp. 1 – 15, 2011.

[19] Alan Chamberlain and Andy Crabtree, "Searching for music: understanding the discovery, acquisition, processing and organization of music in a domestic setting for design," *Personal and Ubiquitous Computing*, vol. 20, no. 4, pp. 559–571, Aug 2016.

[20] Frederic Font, S. Oramas, G. Fazekas, and Xavier Serra, "Extending tagging ontologies with domain specific knowledge," in *International Semantic Web Conference*, Riva del Garda, Italy, October 2014, pp. 209–212.

[21] Pasi Saari, Gyorgy Fazekas, Tuomas Eerola, Mathieu Barthet, Olivier Lartillot, and Mark Sandler, "Genre-adaptive semantic computing and audio-based modelling for music mood annotation," *IEEE Transactions on Affective Computing*, vol. 7, no. 2, pp. 122–135, 2016.

[22] Yoav Freund and Robert E Schapire, "Experiments with a new boosting algorithm," in *ICML*, 1996, vol. 96, pp. 148–156.

[23] Chris Baume, György Fazekas, Mathieu Barthet, David Marston, and Mark Sandler, "Selection of audio features for music emotion recognition using production music," in *53rd International Conference of the Audio Engineering Society on Semantic Audio*, Jan 2014.

[24] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, "Content-based music information retrieval: Current directions and future challenges," *Proceedings of the IEEE*, vol. 96, no. 4, pp. 668–696, April 2008.

[25] Matthew D Hoffman, David M Blei, and Perry R Cook, "Content-based musical similarity computation using the hierarchical dirichlet process.," in

ISMIR, 2008, pp. 349–354.

[26] Florian Eyben, Sebastian Böck, Björn W Schuller, and Alex Graves, "Universal onset detection with bidirectional long short-term memory neural networks.," in *Ismir*, 2010, pp. 589–594.

[27] Philippe Hamel and Douglas Eck, "Learning features from music audio with deep belief networks.," in *ISMIR*. Utrecht, The Netherlands, 2010, pp. 339–344.

[28] Xinquan Zhou and Alexander Lerch, "Chord detection using deep learning," in *Proceedings of the 16th ISMIR Conference*, 2015, vol. 53.

[29] Jan Schlüter and Thomas Grill, "Exploring data augmentation for improved singing voice detection with neural networks," *ISMIR*, 2015.

[30] Siddharth Sigtia, Nicolas Boulanger-Lewandowski, and Simon Dixon, "Audio chord recognition with a hybrid recurrent neural network.," in *ISMIR*, 2015, pp. 127–133.

[31] Dawen Liang, Minshu Zhan, and Daniel PW Ellis, "Content-aware collaborative music recommendation using pre-trained neural networks," in *Conference of the International Society for Music Information Retrieval (ISMIR 2015)*. Malaga, Spain, 2015, pp. 295–301.

[32] Thomas Grill and Jan Schlüter, "Music boundary detection using neural networks on spectrograms and self-similarity lag matrices," in *Proceedings of the 23rd European Signal Processing Conference (EUSPICO 2015), Nice, France*, 2015.

[33] Sebastian Böck, Florian Krebs, and Gerhard Widmer, "Accurate tempo estimation based on recurrent neural networks and resonating comb filters.," in *ISMIR*, 2015, pp. 625–631.

[34] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams, "Learning internal representations by error propagation," Tech. Rep., DTIC Document, 1985.

[35] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E

Howard, Wayne Hubbard, and Lawrence D Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[36] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[37] Xavier Glorot, Antoine Bordes, and Yoshua Bengio, "Deep sparse rectifier neural networks.," in *Aistats*, 2011, vol. 15, p. 275.

[38] George E Dahl, Tara N Sainath, and Geoffrey E Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8609–8613.

[39] Matthew D Zeiler, M Ranzato, Rajat Monga, Min Mao, Kun Yang, Quoc Viet Le, Patrick Nguyen, Alan Senior, Vincent Vanhoucke, Jeffrey Dean, et al., "On rectified linear units for speech processing," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3517–3521.

[40] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[41] Sebastian Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[42] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, MIT press, 2016.

[43] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[45] Yoshua Bengio, Patrice Simard, and Paolo Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural*

networks, vol. 5, no. 2, pp. 157–166, 1994.

[46] Vincent Lostanlen and Carmine-Emanuele Cella, "Deep convolutional networks on the pitch spiral for musical instrument recognition," *Proceedings of the International Society for Music Information Retrieval (ISMIR)*, 2016.

[47] R.M. Bittner, B. McFee, J. Salamon, P. Li, and J.P. Bello, "Deep salience representations for f_0 estimation in polyphonic music," in *18th International Society for Music Information Retrieval Conference*, 2017, ISMIR.

[48] Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis, "Singing-voice separation from monaural recordings using deep recurrent neural networks.," in *ISMIR*, 2014, pp. 477–482.

[49] Brian CJ Moore, *An introduction to the psychology of hearing*, Brill, 2012.

[50] Junichi Yamagishi and Simon King, "Simple methods for improving speaker-similarity of hmm-based speech synthesis," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, 2010, pp. 4610–4613.

[51] Ben J Shannon and Kuldip K Paliwal, "A comparative study of filter bank spacing for speech recognition," in *Microelectronic engineering research conference*, 2003, vol. 41.

[52] Douglas O'shaughnessy, *Speech communication: human and machine*, Universities press, 1987.

[53] Sander Dieleman and Benjamin Schrauwen, "Multiscale approaches to music audio feature learning.," in *Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR 2013, Curitiba, Brazil, November 4-8, 2013*, 2013.

[54] Karen Ullrich, Jan Schlüter, and Thomas Grill, "Boundary detection in music structure analysis using convolutional neural networks," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014), Taipei, Taiwan*, 2014.

[55] Jan Schluter and Sebastian Bock, "Improved musical onset detection with

convolutional neural networks," in *Acoustics, Speech and Signal Processing, IEEE International Conference on*. IEEE, 2014.

[56] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen, "Deep content-based music recommendation," in *Advances in Neural Information Processing Systems*, 2013, pp. 2643–2651.

[57] Eric J Humphrey and Juan P Bello, "Rethinking automatic chord recognition with convolutional neural networks," in *Machine Learning and Applications (ICMLA), International Conference on*. IEEE, 2012.

[58] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon, "An end-to-end neural network for polyphonic music transcription," *arXiv preprint arXiv:1508.01774*, 2015.

[59] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer, "On the potential of simple framewise approaches to piano transcription," in *International Society of Music Information Retrieval (ISMIR), New York, USA*, 2016.

[60] Juan Pablo Bello, "Chroma and tonality," *Music Information Retrieval Course*.

[61] Takuya Fujishima, "Realtime chord recognition of musical sound: a system using common lisp music.," in *ICMC*, 1999, pp. 464–467.

[62] Gregory H Wakefield, "Mathematical representation of joint time-chroma distributions," in *SPIE's International Symposium on Optical Science, Engineering, and Instrumentation*. International Society for Optics and Photonics, 1999, pp. 637–645.

[63] Siddharth Sigtia and Simon Dixon, "Improved music feature learning with deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6959–6963.

[64] Philippe Hamel, Matthew EP Davies, Kazuyoshi Yoshii, and Masataka Goto, "Transfer learning in mir: Sharing learned latent representations for music audio classification and similarity," *14th International Conference on Music*

Information Retrieval, 2013.

[65] Filip Korzeniowski and Gerhard Widmer, "Feature learning for chord recognition: The deep chroma extractor," *arXiv preprint arXiv:1612.05065*, 2016.

[66] Stefan Uhlich, Franck Giron, and Yuki Mitsufuji, "Deep neural network based instrument extraction from music," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2135–2139.

[67] Junqi Deng and Yu-Kwong Kwok, "Automatic chord estimation on seventhsbass chord vocabulary using deep neural network," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 261–265.

[68] Simon Durand, Juan P Bello, Bertrand David, and Gaël Richard, "Downbeat tracking with multiple features and deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 409–413.

[69] François Rigaud and Mathieu Radenen, "Singing voice melody transcription using deep neural networks," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR). New York*, 2016, pp. 737–743.

[70] Il-Young Jeong and Kyogu Lee, "Learning temporal features using a deep neural network and its application to music genre classification," *Proc. of the 17th Int. Society for Music Information Retrieval Conf.(ISMIR)*, 2016.

[71] Min Lin, Qiang Chen, and Shuicheng Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2013.

[72] Yusuf Aytar, Carl Vondrick, and Antonio Torralba, "Soundnet: Learning sound representations from unlabeled video," in *Advances in Neural Information Processing Systems*, 2016, pp. 892–900.

[73] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Con-*

ference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[74] Li-Chia Yang, Szu-Yu Chou, Jen-Yu Liu, Yi-Hsuan Yang, and Yi-An Chen, "Revisiting the problem of audio-based hit song prediction using convolutional neural networks," *arXiv preprint arXiv:1704.01280*, 2017.

[75] Jordi Pons, Olga Slizovskaia, Rong Gong, Emilia Gómez, and Xavier Serra, "Timbre analysis of music audio signals with convolutional neural networks," 2017.

[76] Jordi Pons and Xavier Serra, "Designing efficient architectures for modeling temporal features with convolutional neural networks," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2017.

[77] Jongpil Lee, Jiyoung Park, Keunhyoung Luke Kim, and Juhan Nam, "Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms," *arXiv preprint arXiv:1703.01789*, 2017.

[78] Tara N Sainath, Ron J Weiss, Andrew Senior, Kevin W Wilson, and Oriol Vinyals, "Learning the speech front-end with raw waveform cldnns," in *Proc. Interspeech*, 2015.

[79] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in neural information processing systems*, 2009, pp. 1096–1104.

[80] Lihua Li, "Audio musical genre classification using convolutional neural networks and pitch and tempo transformations," 2010.

[81] Tom LH Li, Antoni B Chan, and A Chun, "Automatic musical pattern feature extraction using convolutional neural network," in *Proc. Int. Conf. Data Mining and Applications*, 2010.

[82] Eric J Humphrey and Juan P Bello, "From music audio to chord tablature: Teaching deep convolutional networks toplay guitar," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on.* IEEE, 2014, pp. 6974–6978.

[83] Jan Wülfing and Martin Riedmiller, "Unsupervised learning of local features for music classification.," in *International Society of Music Information Retrieval Conference*. ISMIR, 2012, pp. 139–144.

[84] Jan Schlüter and Sebastian Böck, "Musical onset detection with convolutional neural networks," in *6th International Workshop on Machine Learning and Music (MML), Prague, Czech Republic*, 2013.

[85] Rui Lu, Kailun Wu, Zhiyao Duan, and Changshui Zhang, "Deep ranking: triplet matchnet for music metric learning," 2017.

[86] Yoonchang Han, Jaehun Kim, and Kyogu Lee, "Deep convolutional neural networks for predominant instrument recognition in polyphonic music," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 1, pp. 208–221, 2017.

[87] Andrew JR Simpson, Gerard Roma, and Mark D Plumbley, "Deep karaoke: Extracting vocals from musical mixtures using a convolutional deep neural network," *arXiv preprint arXiv:1504.04658*, 2015.

[88] Jan Schlüter, "Learning to pinpoint singing voice from weakly labeled examples," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 44–50.

[89] Merlijn Blaauw and Jordi Bonada, "A neural parametric singing synthesizer," *arXiv preprint arXiv:1704.03809*, 2017.

[90] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "Wavenet: A generative model for raw audio," *CoRR abs/1609.03499*, 2016.

[91] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi, "Neural audio synthesis of musical notes with wavenet autoencoders," *arXiv preprint arXiv:1704.01279*, 2017.

[92] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, "Learning phrase representations using

rnn encoder-decoder for statistical machine translation," in *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014.

[93] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR 2015*, 2014.

[94] Haşim Sak, Andrew Senior, and Françoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[95] Simon Leglaive, Romain Hennequin, and Roland Badeau, "Singing voice detection with deep recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 121–125.

[96] Mike Schuster and Kuldip K Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[97] Siddharth Sigtia, Emmanouil Benetos, Nicolas Boulanger-Lewandowski, Tillman Weyde, Artur S d'Avila Garcez, and Simon Dixon, "A hybrid recurrent neural network for music transcription," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2061–2065.

[98] Xinxing Li, Haishu Xianyu, Jiashen Tian, Wenxiao Chen, Fanhang Meng, Mingxing Xu, and Lianhong Cai, "A deep bidirectional long short-term memory based multi-scale approach for music dynamic emotion prediction," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 544–548.

[99] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[100] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter, "Fast and

accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[101] Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet, "Towards musical query-by-semantic-description using the cal500 data set," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 439–446.

[102] Edith Law, Kris West, Michael I Mandel, Mert Bay, and J Stephen Downie, "Evaluation of algorithms using games: The case of music tagging.," in *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, Kobe, Japan, October 26-30, 2009*. 2009, pp. 387–392, ISMIR.

[103] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere, "The million song dataset," in *ISMIR 2011: Proceedings of the 12th International Society for Music Information Retrieval Conference, October 24-28, 2011, Miami, Florida*. University of Miami, 2011, pp. 591–596.

[104] Benoît Frénay and Michel Verleysen, "Classification in the presence of label noise: a survey," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 5, pp. 845–869, 2014.

[105] Volodymyr Mnih and Geoffrey E Hinton, "Learning to label aerial images from noisy data," in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, 2012, pp. 567–574.

[106] Sainbayar Sukhbaatar and Rob Fergus, "Learning from noisy labels with deep neural networks," *arXiv preprint arXiv:1406.2080*, vol. 2, no. 3, pp. 4, 2014.

[107] Derek Tingle, Youngmoo E Kim, and Douglas Turnbull, "Exploring automatic music annotation with acoustically-objective tags," in *Proceedings of the international conference on Multimedia information retrieval*. ACM, 2010, pp. 55–62.

[108] Christopher Z Mooney and Robert D Duval, *Bootstrapping: A nonparametric*

approach to statistical inference, Number 94-95. Sage, 1993.

[109] Lorenzo Torresani, "Weakly supervised learning," in *Computer Vision*, pp. 883–885. Springer, 2014.

[110] Duyu Tang, Bing Qin, and Ting Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1422–1432.

[111] Zhen Zuo, Bing Shuai, Gang Wang, Xiao Liu, Xingxing Wang, Bing Wang, and Yushi Chen, "Convolutional recurrent neural networks: Learning spatial dependencies for image representation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 18–26.

[112] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, 2016.

[113] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[114] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[115] François Chollet, "Keras," *GitHub repository: https://github. com/fchollet/keras*, 2015.

[116] The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, et al., "Theano: A python framework for fast computation of mathematical expressions," *arXiv preprint arXiv:1605.02688*, 2016.

[117] Kurt Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[118] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

[119] Brian McFee, Matt McVicar, Colin Raffel, Dawen Liang, Oriol Nieto, Eric Battenberg, Josh Moore, Dan Ellis, Ryuichi YAMAMOTO, Rachel Bittner, Douglas Repetto, Petr Viktorin, João Felipe Santos, and Adrian Holovaty, "librosa: 0.4.1," Oct. 2015.

[120] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[121] Malcolm Slaney, "Auditory toolbox," *Interval Research Corporation, Tech. Rep*, vol. 10, pp. 1998, 1998.

[122] Samuel F. Dodge and Lina J. Karam, "Understanding how image quality affects deep neural networks," *CoRR*, vol. abs/1604.04004, 2016.

[123] Jordi Pons, Oriol Nieto, Matthew Prockup, Erik M Schmidt, Andreas F Ehmann, and Xavier Serra, "End-to-end learning for music audio tagging at scale," *arXiv preprint arXiv:1711.02520*, 2017.

[124] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1717–1724.

[125] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[126] Quoc V Le and Tomas Mikolov, "Distributed representations of sentences and documents.," in *International Conference on Machine Learning*, 2014,

vol. 14, pp. 1188–1196.

[127] Jason Weston, Samy Bengio, and Nicolas Usunier, "Wsabie: Scaling up to large vocabulary image annotation," *International Joint Conference on Artificial Intelligence*, 2011.

[128] Adam Coates and Andrew Y Ng, "Learning feature representations with k-means," in *Neural Networks: Tricks of the Trade*, pp. 561–580. Springer, 2012.

[129] Aäron Van Den Oord, Sander Dieleman, and Benjamin Schrauwen, "Transfer learning by supervised pre-training for audio-based music classification," in *Conference of the International Society for Music Information Retrieval (ISMIR 2014)*, 2014.

[130] Keunwoo Choi, György Fazekas, and Mark Sandler, "Towards playlist generation algorithms using rnns trained on within-track transitions," in *Workshop on Surprise, Opposition, and Obstruction in Adaptive and Personalized Systems (SOAP), Halifax, Canada, 2016*, 2016.

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

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

[133] Min Lin, Qiang Chen, and Shuicheng Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[134] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*. IEEE, 2004, vol. 2, pp. 985–990.

[135] Yujun Zeng, Xin Xu, Yuqiang Fang, and Kun Zhao, "Traffic sign recognition using extreme learning classifier with deep convolutional features," in *The*

2015 international conference on intelligence science and big data engineering (IScIDE 2015), Suzhou, China, 2015.

[136] Grzegorz Gwardys and Daniel Grzywczak, "Deep image features in music information retrieval," *International Journal of Electronics and Telecommunications*, vol. 60, no. 4, pp. 321–326, 2014.

[137] Johan AK Suykens and Joos Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[138] Alex J Smola and Bernhard Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[139] Ugo Marchand and Geoffroy Peeters, "The extended ballroom dataset," *Conference of the International Society for Music Information Retrieval (ISMIR 2016) late-breaking session*, 2016.

[140] George Tzanetakis and Perry Cook, "Musical genre classification of audio signals," *Speech and Audio Processing, IEEE transactions on*, vol. 10, no. 5, pp. 293–302, 2002.

[141] George Tzanetakis, "Gtzan musicspeech," *availabe online at http://marsyas.info/download/data sets*, 1999.

[142] Mohammad Soleymani, Micheal N Caro, Erik M Schmidt, Cheng-Ya Sha, and Yi-Hsuan Yang, "1000 songs for emotional analysis of music," in *Proceedings of the 2nd ACM international workshop on Crowdsourcing for multimedia*. ACM, 2013, pp. 1–6.

[143] Mathieu Ramona, Gaël Richard, and Bertrand David, "Vocal detection in music with support vector machines," in *Acoustics, Speech and Signal Processing (ICASSP), 2008 IEEE International Conference on*, March 31 - April 4 2008, pp. 1885–1888.

[144] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *22st ACM International Conference on Multimedia (ACM-MM'14)*, Orlando, FL, USA, Nov. 2014.

[145] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

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

[147] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[148] Bob L Sturm et al., "Revisiting priorities: Improving mir evaluation practices," in *Proc. 17th International Society for Music Information Retrieval Conference (ISMIR'16), New York, NY, USA*, 2016.

[149] Ugo Marchand and Geoffroy Peeters, "Scale and shift invariant time/frequency representation using auditory statistics: Application to rhythm description," in *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on*. IEEE, 2016, pp. 1–6.

[150] Fabien Gouyon, Simon Dixon, Elias Pampalk, and Gerhard Widmer, "Evaluating rhythmic descriptors for musical genre classification," in *25th AES International Conference, London, UK*, 2004.

[151] Athanasios Lykartsis and Alexander Lerch, "Beat histogram features for rhythm-based musical genre classification using multiple novelty functions," in *Proceedings of the 16th ISMIR Conference*, 2015, pp. 434–440.

[152] Aggelos Gkiokas, Vassilis Katsouros, and George Carayannis, "Towards multi-purpose spectral rhythm features: An application to dance style, meter and tempo estimation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 11, pp. 1885–1896, 2016.

[153] Arash Foroughmand Arabi and Guojun Lu, "Enhanced polyphonic music genre classification using high level features," in *Signal and Image Processing Applications (ICSIPA), 2009 IEEE International Conference on*. IEEE,

2009, pp. 101–106.

[154] Babu Kaji Baniya, Joonwhoan Lee, and Ze-Nian Li, "Audio feature reduction and analysis for automatic music genre classification," in *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*. IEEE, 2014, pp. 457–462.

[155] Yin-Fu Huang, Sheng-Min Lin, Huan-Yu Wu, and Yu-Siou Li, "Music genre classification based on local feature selection using a self-adaptive harmony search algorithm," *Data & Knowledge Engineering*, vol. 92, pp. 60–76, 2014.

[156] Hanchuan Peng, Fuhui Long, and Chris Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[157] Chia-Ming Wang and Yin-Fu Huang, "Self-adaptive harmony search algorithm for optimization," *Expert Systems with Applications*, vol. 37, no. 4, pp. 2826–2837, 2010.

[158] M Srinivas, Debaditya Roy, and C Krishna Mohan, "Learning sparse dictionaries for music and speech classification," in *Digital Signal Processing (DSP), 2014 19th International Conference on*. IEEE, 2014, pp. 673–675.

[159] Felix Weninger, Florian Eyben, and Bjorn Schuller, "On-line continuous-time music mood regression with deep recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 5412–5416.

[160] Justin Salamon and Juan Pablo Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, 2017.

[161] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th Python in Science Conference*, 2015.

[162] Jason Jo and Yoshua Bengio, "Measuring the tendency of cnns to learn

surface statistical regularities," *CoRR*, vol. abs/1711.11561, 2017.

[163] Yeonwoo Jeong, Keunwoo Choi, and Hosan Jeong, "Dlr: Toward a deep learned rhythmic representation for music content analysis," *arXiv preprint arXiv:1712.05119*, 2017.

[164] Sergio Oramas, Oriol Nieto, Francesco Barbieri, and Xavier Serra, "Multi-label music genre classification from audio, text, and images using deep features," *arXiv preprint arXiv:1707.04916*, 2017.