공학박사 학위논문

특징 학습을 통한 악기 식별과 음색 분류

Musical Instrument Identification and
Tone Detection Using Feature Learning

2017년 2월

서울대학교 융합과학기술대학원

융합과학부 디지털정보융합전공

한윤창

공학박사 학위논문

특징 학습을 통한 악기 식별과 음색 분류

Musical Instrument Identification and
Tone Detection Using Feature Learning

2017년 2월

서울대학교 융합과학기술대학원

융합과학부 디지털정보융합전공

한윤창

특징 학습을 통한 악기 식별과 음색 분류

Musical Instrument Identification and Tone Detection Using Feature Learning

지도교수 이교구

이 논문을 공학박사 학위논문으로 제출함

2017 년 1 월

서울대학교 융합과학기술대학원

융합과학부 디지털정보융합전공

한윤창

한윤창의 공학박사 학위논문을 인준함

2017 년 1 월

위 원 장	이 원 종
부위원장	이 교 구
위 원	곽 노 준
위 원	남 주 한
위 원	서 봉 원

Abstract

In music information retrieval (MIR) field, most of the tasks have been heavily relying on the hand-crafted features which are highly useful to measure and quantify the certain target characteristics of the sound such as pitch, roughness, and brightness. However, there are an increasing number of attempts on applying feature learning technique, which has shown superior performance across the research fields, in MIR tasks especially when the goal is the identification of the sound. The aim of this thesis is to advance the state-of-the-art in musical instrument identification and tone detection tasks with feature learning approaches, which can be used for various music-related applications, including but not limited to, music searching, browsing, recommendation, and education. We utilize sparse feature learning and convolutional neural network to learn a feature from input data, and propose the network architecture and data processing framework that is suitable for music signal. We present experimental results on MIR tasks such as fingering detection of overblown flute sound, instrument identification in monophonic sound, and predominant instrument recognition in a real-world polyphonic music. In addition, we conducted an extensive experiment to find the optimal data processing techniques and parameters such as input frame sampling, frequency scaling, activation pooling, window/hop size, output aggregation method, and network training hyperparameters.

Keywords: music information retrieval, deep learning, convolutional neural network, sparse feature learning, musical instrument identification
Student Number: 2011-31242

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Music is humanly organized sound, organized with intent, into a recognizable aesthetic entity [1]. Human society has been always with music for various reasons such as communication, culture, religion, or musical pleasure, and there is almost uncountable number of music exists in the world.

The rapid development of technology has changed the way people listen to the music. Nowadays, people can easily buy music on the internet, or simply listen to music through online streaming services. This increasing consumption of music in a digital format greatly improved the accessibility of the people on the music compare to buying albums in a physical format such as tapes and CDs. However, searching and browsing of the music still mostly rely on the metadata enclosed in the audio files which contain information such as title, album name, release year, track number, and genre as illustrated in Fig. 1.1. Although metadata conveys a useful information about the music, information fields of the metadata are often empty because it is not a compulsory to include them in an audio file. In addition, more importantly, there is no information

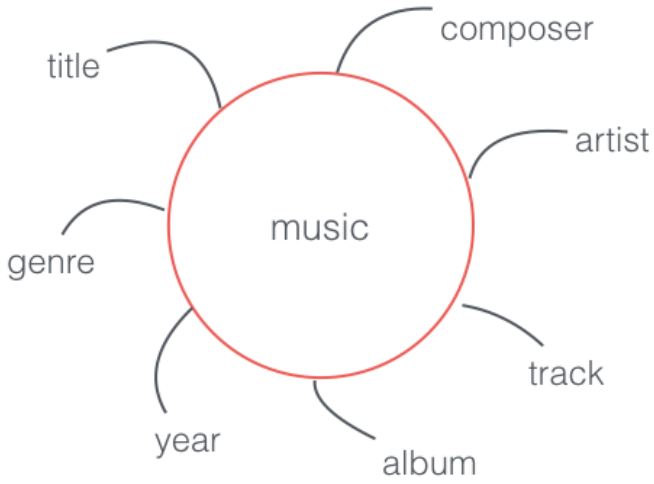

Fig. 1.1 ID3 metadata enclosed in mp3 format audio file format. It contains limited information about music content itself, also information fields are often empty or broken.

about the music content itself except genre in the metadata.

Music information retrieval (MIR) is a research field that focuses on extracting higher-level information from the music using computational methods such as digital signal processing and machine learning techniques along with the musical domain knowledge. We begin with introducing various research fields of music information retrieval. Then, we explain the target problems we aim to solve in the thesis and clarify the contribution of this thesis.

1.1 Music Information Retrieval

There is various information included in the music, and as mentioned above, MIR research aims to extract higher level information from the music which is not visible in their raw form as shown in Fig. 1.2. Although there are no

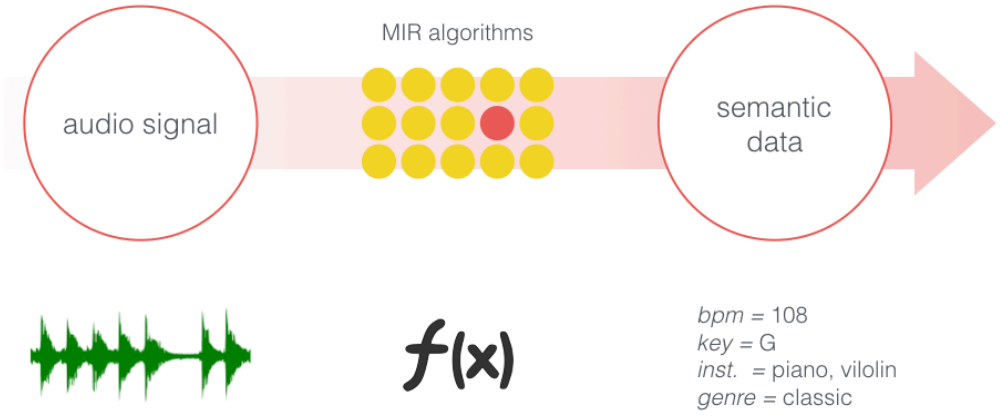

Fig. 1.2 MIR algorithms aims to extract higher level semantic information from the music.

clear boundaries among MIR tasks, Music Information Retrieval Evaluation eXchange (MIREX) is a good starting point to observe different types of music information. MIREX is a satellite event of The International Society of Music Information Retrieval (ISMIR) conference, which is one of the largest MIR conferences, and they provide framework and data to evaluate MIR algorithms. In 2016, they set 24 MIR algorithm evaluation tasks and full list the tasks is available in their website [1]. To name a few, they evaluate tasks including genre/mood classification, onset/key detection, beat tracking, tempo/downbeat estimation, query by humming/tapping, music fingerprinting, melody extraction, score alignment, and structure segmentation. Apart from the tasks included in the list, there are many more important topics exist such as pitch estimation, instrument identification, and source separation.

The higher level information such as mood, genre, and tempo can be used for searching and browsing the music. For instance, people can search music with specific conditions such as genre or mood. In addition, it also allows

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

content-based music recommendation which currently heavily rely on the user behavioral data only. For example, if someone liked to listen to a certain music, then the service provider can recommend similar music, also it is possible to suggest a song with the suitable mood depending on a context such as time and weather. In addition, music fingerprinting is one of commercially successful MIR technologies. Music searching services such as Shazam[2] and Soundhound[3] use music fingerprinting technology, and people can find the title and artist of the music by simply turn on the smartphone application. There are not many commercially successful examples that use query by humming and tapping, but it has great potential to open up a new way of searching music once they achieve reasonably good accuracy.

MIR technologies have changed not only how people search the music, but also how people learn and play the music instrument. There are many smartphone and tablet applications available for music education, which evaluates how the performer played the instrument. Using onset and pitch detection, it is possible to give a feedback about the mistakes, and score following technology can automatically turn the pages of the music score for musicians during a performance. In addition, beat tracking and tempo estimation are already widely implemented in digital disk jockey (DJ) equipment. Many DJ programs and equipment display beat per minute (BPM) information of the songs to help music selection. Also, they also help accurate beat sync between the currently played song and the next song to minimize the mistake.

Moreover, MIR technologies often used as a pre-processing step for other MIR algorithms to improve the performance further. For instance, harmonic-percussive source separation technique frequently performed prior to pitch de-

[2] `https://www.shazam.com/`
[3] `http://soundhound.com/`

tection or melody extraction algorithm to minimize the effect of drum sound, and beat tracking algorithm is often used in a feature extraction step of the music structure segmentation algorithm to extract the beat-synchronous features.

As shown above, there are various MIR topics being researched to extract semantic information from the music. Among this music information, we focus on investigating identification of the instrument sound. In the next section, we explain more details about musical instrument identification and tone detection as well as its importance and possible real-world applications.

1.2 Musical Instrument Identification and Tone Detection

Music can be said to be built by the interplay of various instruments. A human can easily identify what instruments are used in a music, but it is still a difficult task for a computer to automatically recognize them. This is mainly because music in the real-world is mostly polyphonic, which makes extraction of information from an audio highly challenging. Furthermore, instrument sounds in the real-world vary in many ways such as for timbre, quality, and playing style, which makes identification of the musical instrument even harder.

In the MIR field, it is highly desirable to know what instruments are used in an audio sample because these can be useful in many ways. First of all, instrument information per se is an important and useful information for users, and it can be included in the audio tags. There is a huge demand for music search owing to the increasing number of music files in digital format. Unlike text search, it is difficult to search for music because input queries are usually in text format. If an instrument information is included in the tags, it allows people to search for music with the specific instrument they want.

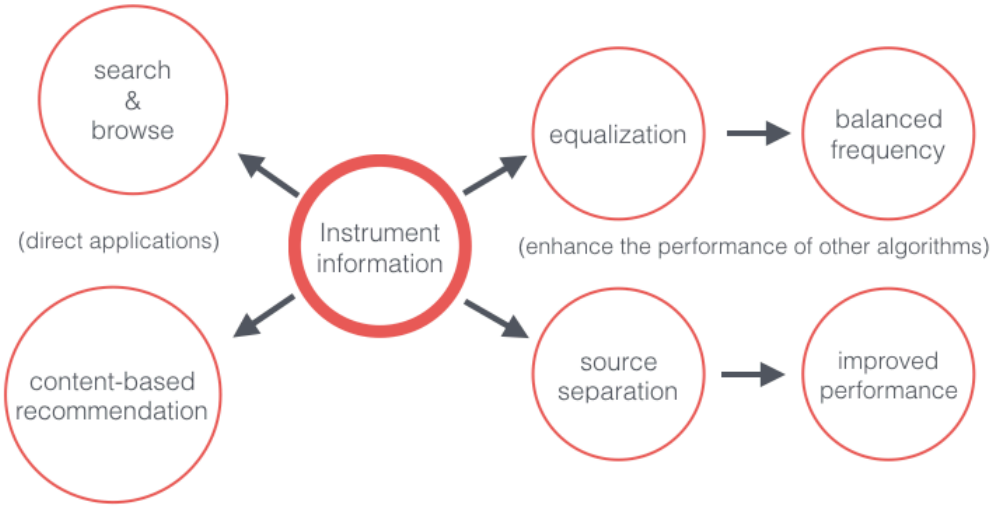

Fig. 1.3 Possible applications of instrument information. It can be used for various application directly, also can be used for enhance the performance of other MIR algorithms.

In addition, the obtained instrument information can be used for various audio/music applications as illustrated in Fig. 1.3 For instance, more instrument-specific and tailored audio equalization can be applied to the music; moreover, a music recommendation system can reflect the preference of users for musical instruments. Furthermore, it can also be used to enhance the performance of other MIR tasks. For example, knowing the number and type of the instrument would significantly improve the performance of source separation and automatic music transcription; it would also be helpful for identifying the genre of the music because the instrument information is an important clue for differentiating the music genre.

The identifying musical instrument can be done because all the instruments have its own unique timbre. However, this timbral difference also exists within the sound generated from a single instrument. It is because musical instrument can be played with various tones using different playing styles and techniques.

For instance, a flute can be played to produce more 'airy' sound on purpose by using altering fingering and blowing pressure. Similarly, a piano can be played with more 'dull' sound by pressing the pedal. This kind of timbral difference is minor compared to the difference between instruments, but it is still very important because affects the musical 'feeling' of the performance. In the next section, we present existing approaches on instrument identification and tone detection tasks.

1.3 Related Works

Instrument identification can be performed in various forms. Hence, the term 'instrument identification' (or recognition) might indicate several different research topics. For instance, many of the related works focus on studio-recorded isolated notes.

To name a few, Kitahara et al. proposed an F0-dependent multivariate normal distribution with a discriminant function based on the Bayes decision rule [2]. They used a set of 129 features including spectral, temporal, modulation, and inharmonic component features, and their experimental result using 6247 solo tones of 19 musical instruments achieved 79.73%. Next, Eronen and Klapuri's work used a set of 43 cepstral coefficients and temporal features [3]. The system achieved 80.6% accuracy for identifying 1498 solo tones from 30 individual orchestral instruments. This paper reports that there is no significant advantage when using the instrument family hierarchy for identification performance. Diment et al. used a modified group delay feature that incorporates phase information together with mel-frequency cepstral coefficients (MFCCs) and achieved a classification accuracy of about 71% for 22 instruments [4].

One of the most recent works is that by Bhalke et al., in which high-order

spectrum (bi/tri-spectrum) features are applied along with a set of conventional spectral features for identifying 760 solo tones of 19 musical instruments [5]. They used a multilayered feed-forward neural network as a classifier and obtained an accuracy of 88.57% for individual instrument recognition.

Another interesting approach for identifying isolated tones is Newton and Smith's method using a neurally inspired sound onset fingerprint [6]. They used a gammatone filterbank to model the first-order response of a basilar membrane; then, the output of each filter was spike-encoded to simulate a human auditory nerve. The system achieved an accuracy of 75.6% for identifying 2085 solo tones of five instrument categories and was not tested for an individual instrument.

Some previous works such as Krishna and Sreenivas [7] experimented with a classification for solo phrases rather than for isolated notes. They proposed line spectral frequencies (LSF) with a Gaussian mixture model (GMM) and achieved an accuracy of about 77% for instrument family and 84% for 14 individual instruments. Moreover, Essid et al. [8] reported that a classification system with MFCCs and GMM along with principal components analysis (PCA) achieved an overall recognition accuracy of about 67% on solo phrases with five instruments.

More recent works deal with polyphonic sound, which is closer to real-world music than to monophonic sound. In the case of polyphonic sound, a number of research studies used synthesized polyphonic audio from studio-recorded single tones. Heittola et al. [9] used a non-negative matrix factorization (NMF)-based source-filter model with MFCCs and GMM for synthesized polyphonic sound and achieved a recognition rate of 59% for six polyphonic notes randomly generated from 19 instruments. Kitahara et al. [10] used various spectral, temporal, and modulation features with PCA and linear discriminant analysis (LDA) for classification. They reported that using feature weighting and musical context, recognition rates were about 84% for a duo, 78% for a trio, and 72% for a

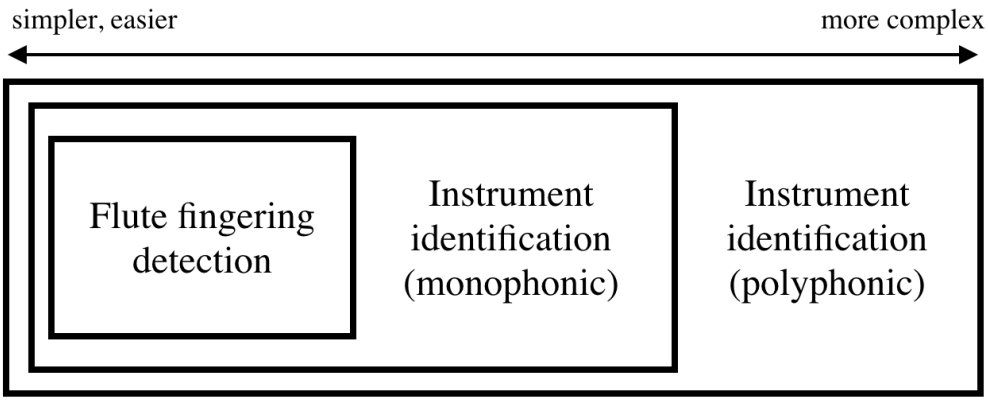

Fig. 1.4 Three main research topics of the thesis, beginning from relatively simple flute fingering detection and expand to instrument identification in monophonic and polyphonic sound.

quartet. Duan et al. [11] proposed the uniform discrete cepstrum (UDC) and mel-scale UDC (MUDC) as a spectral representation with a radial basis function (RBF) kernel support vector machine (SVM) to classify 13 types of Western instruments. The classification accuracy of randomly mixed chords of two and six polyphonic notes, generated using isolated note samples from the RWC musical instrument sound database [12], was around 37% for two polyphony notes and 25% for six polyphony notes.

As shown above, instrument identification and tone detection can be performed in a various way depending on input data type and target application. In the following section, we explain the target tasks we aim to solve in this thesis.

1.4 Tasks of Interest

Instrument identification and tone detection can be done in many different ways depending on input data types and the target application. In terms of the input

source, various types of input sources can be used to solve these tasks. It can be, including but not limited to, recorded sound, synthesized sound, or symbolic data such as Musical Instrument Digital Interface (MIDI) for music analysis. Usually, extracting information from the recorded sound is much harder than analyzing synthesized sound or symbolic data due to the effect of noise and recording environment. However, an increasing number recent MIR researches focus on extracting information from the recorded sound because most of the real-world musical sound are recorded sound, and it is much more meaningful in terms of application possibility. In this thesis, we limit the input data of the system as recorded audio to observe the potential of developed algorithms for the real-world implementation. We aim to identify the instrument tone in three different levels, beginning with a relatively simpler case and expand to the more complex situation as shown in Fig. 1.4.

Firstly, we experiment with tone detection for a specific instrument, flute in this case. We demonstrate a method to differentiate overblown flute sound and normally blown sound to estimate what fingering is used for the sound production. The 'overblowing' is widely used technique for a various woodwind instrument which is done by altering blowing pressure and fingering to add special 'color' and 'feeling' on the performance. Usually, the overblown tone is described as 'airy' tone, but its minor timbral difference is hard to recognize for non-musicians or beginner flute performers. In this research, we aim to classify timbral difference among sounds from different fingerings with the same pitch. Once this can be identified, it can be used for adding fingering information for music transcription which typically only detects pitch and onset. In addition, it allows music education application for beginners to detect unintentional overblowing sound caused by inappropriate fingering or blowing pressure.

Next, we experiment with the identification of solo tones of various instru-

ments. Although the timbral difference between instruments is larger than that of within single instrument, in this research, we aim to identify the sound of instruments with various playing styles, articulations, manufacturers, and performers. Also, total 24 instruments are used for the experiment including highly similar instruments in terms of timbre such as violin, viola, and cello. Both of flute overblown sound detection and instrument identification experiments explained above are done under the monophonic situation which means there is no overlap between tones.

Finally, we expand our goal to instrument identification in a polyphonic situation. In this research, we investigate a method to detect predominant instruments in the music excerpts. The number of target instruments was 11 and all audio clips are extracted from real-world music with various musical instrument types, performers, articulations, production styles, audio quality, and recording environments, which can be considered as nearly identical to a real-world situation. While the first two experiments are a multi-class single-label problem, this experiment assumes that there might exist multiple predominant instruments in the music (i.e., multi-class multi-label problem).

1.5 Contributions

The main contribution of the thesis is proposing a feature learning framework for instrument identification and tone detection. The conventional approach to solve MIR tasks heavily relies on hand-crafted features which require careful feature engineering using domain knowledge on music and signal processing skills. Feature learning aims to automatically learn a suitable feature representation from the input data using machine learning techniques, and this approach has shown superior performance compare to hand-crafted features in various audio

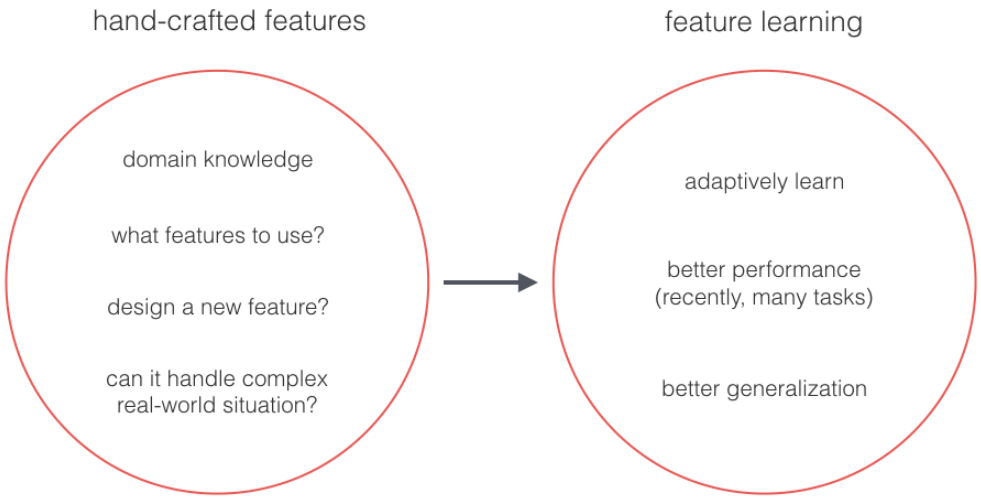

Fig. 1.5 Comparison between hand-crafted features and feature learning.

processing tasks while requiring less domain knowledge. Comparison between hand-crafted features and feature learning is illustrated in Fig. 1.5, and More detailed explanation about various feature representation and learning strategies are explained in the next chapter.

Compare to audio processing field, feature learning techniques are much more heavily employed in computer vision area. Although it aims to learn feature representation automatically, there are many factors to care about to make a feature learning framework works for audio processing as listed in Fig. 1.6.

For example, input representation, data preprocessing, input data organization method, frequency scale, network architecture, learning parameters, output aggregation method, and window/hop size are all highly important for successful feature learning system. There are an increasing number of attempts on using learned feature instead of hand-crafted features in audio processing field, but many parts of the system still need to be improved and investigated. In this thesis, we aim to contribute to developing a feature learning framework

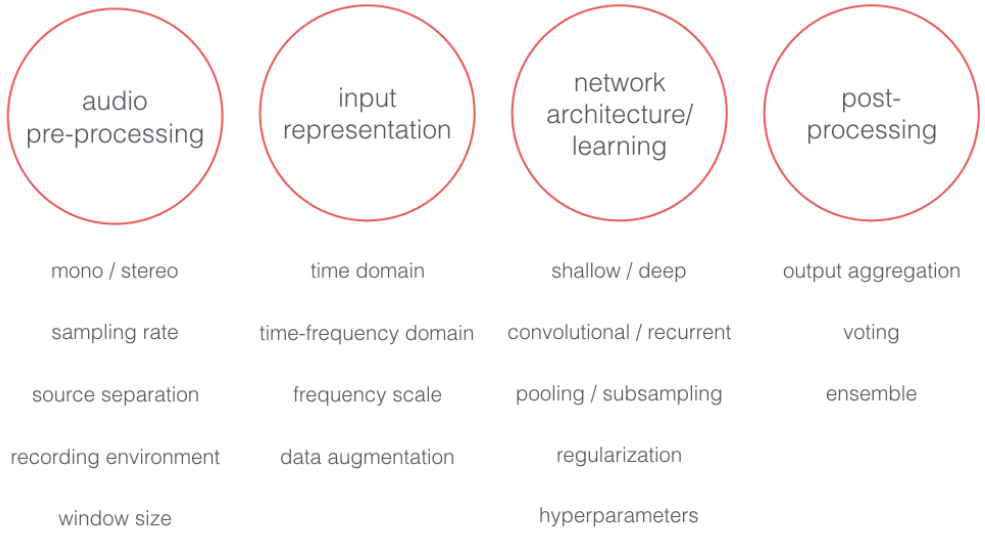

Fig. 1.6 Important factors for successful audio feature learning. Many of them are highly audio-specific and differ from the one for computer vision tasks.

for audio signal processing that can be applied to instrument identification and tone detection, also possibly other MIR and audio processing tasks with small modification as possible. Our major contributions included in each chapter are shown as below.

Chapter 3 We propose a sparse feature learning framework for tone detection to classify the overblown flute sound. Sparse filtering, one of the unsupervised feature learning algorithm, is used for extracting an over-complete set of basis from the input audio. Our system is designed to handle up to sixteenth note at 100 BPM, and suitable frame concatenation and max-pooling size were presented. We visualize learned feature with its activation to observe and understand how it actually works.

Chapter 4 The data processing pipeline used for the instrument identification of monophonic sound is similar to the one that employed in the previous chapter. However, in this chapter, we focus on investigating the optimal frame sampling method, input representation, and activation pooling method for feature learning framework which make a huge difference in performance. For frame sampling, we propose three sampling methods which are fixed-number sampling, proportional sampling, and onset-based sampling. Regarding input representation, we experiment with linear-scale and mel-scale spectrogram to find the optimal frequency scale, and for activation pooling, we compare commonly used max-pooling and average-pooling to proposed standard deviation pooling.

Chapter 5 In this chapter, we focus on developing a convolutional neural network framework for polyphonic instrument identification, and investigate a method to learn features from single-labeled data to estimate the instruments exist in the multi-label music clip. Because the lengths of audio files are usually all different, we analyze input audio with sliding window and aggregate outputs of them to allow the system to handle variable input length. Optimal analysis window size and aggregation method are investigated, also we compare the performance of various activation functions. In addition to average performance, we perform class-wise analysis to observe the performance of each instrument.

As illustrated above, we propose network architecture for each target task which can be a good starting point or hint for the other similar tasks as well. For all the experiments, we present the result of proposed feature learning approach and conventional hand-made features to compare the performance. In addition, we attached qualitative analysis for each experiment to visually analyze the

obtained result, which helps to understand the result and how features are learned through the network. In this chapter, we introduced various research topics in MIR as well as tasks of interest and contribution of the thesis. In the next chapter, we present a typical architecture of MIR systems and explain input data representation, feature extraction, and preprocessing methods which are critical factors for successful MIR systems.

Chapter 2

Overview of MIR Systems for Identification Tasks

There is a wide range of MIR topics being researched to extract various high-level information as illustrated in Section 1.1. Although detailed approaches are all different, general data processing pipelines of modern MIR systems for identification tasks can be seen as composed of several blocks as shown in Figure 2.1.

We divided MIR systems for identification tasks into three large parts which are input data representation, feature representation, and classification. It is possible to observe from the related works presented in Section 1.3 that most

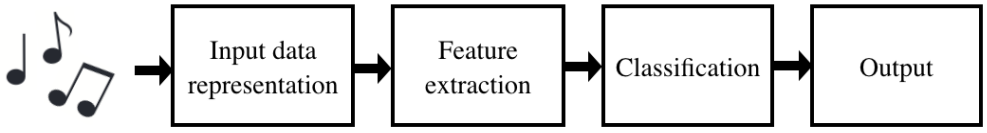

Fig. 2.1 Common data processing architecture of MIR systems for identification tasks.

of the approaches share this general data processing pipeline, In this chapter, we introduce available options for input data representation and feature representation, where the performance improvements are usually made, and explain their advantages and disadvantages of them. Furthermore, we explain input data and feature preprocessing methods for audio processing which are usually briefly mentioned or not included in many papers because it is somewhat clear to the MIR community, but could be a hurdle for researchers from the other fields. Note that we did not include a detailed explanation about classification part, because most of the conventional MIR approaches employ rather general classifiers such as SVM or random forest which is not MIR specific, and neural network approaches usually perform classification step within the network.

2.1 Input Data Representation

Regardless of tasks, input data representation is a highly critical factor for the performance of the system. Depending on the target task, the input signal is transformed to other domain prior to feature extraction step. In this section, we explain time-domain, time-frequency domain, and cepstral domain representation which are the most widely used input data representation for MIR systems.

2.1.1 Time Domain Representation

Sounds are vibrations traveling through the air and recorded using a microphone. This recorded vibration is obviously time domain signal in nature, and it is called waveform. Usage of this raw form of audio signal is highly limited because it only shows the amplitude (i.e., loudness) of the sound. When time domain signal is directly used as an input data for feature extraction, the fea-

ture extraction algorithm should be able to find the patterns exist in the signal. For instance, hand-crafted audio features such as auto-correlation and YIN [13] uses time domain signal to find fundamental frequency that represents pitch of the sound.

Recently, there are an increasing number of attempts on using waveform directly as an input for the convolutional neural network (ConvNet) to learn patterns exist in the signal. This emerging approach is called end-to-end approach, which is promising in that using raw audio signals makes the system rely less on preprocessing and domain knowledge. However, it is reported that end-to-end approach shows slightly lower performance than using time-frequency representation in recent papers [14, 15].

2.1.2 Time-frequency Representation

Audio is usually converted to frequency domain prior to analysis because it gives much more information than the raw time domain audio. Although many sounds are mixed up in the waveform, Fourier transform decomposes the signal into the frequencies that make it up [16]. This frequency domain representation is called spectrum, and discrete Fourier transform (DFT) is defined as:

$$X(k) = \sum_{n=0}^{N-1} x(n) exp(\frac{-jnk2\pi}{N}) \tag{2.1}$$

where $0 \leq n < N-1$ and $0 \leq k < N-1$, also N is transform length and k is frequency bin [17]. In practice, input audio is divided into shorter segments of equal length and DFT is applied, called short-time Fourier transform (STFT), to observe the spectral characteristics of the signal over time.

Although STFT is arguably one of the most popular options for transforming the signal into the time-frequency domain, some MIR researches such as

[18, 19, 20] use discrete wavelet transform (DWT) instead of STFT. DWT provides high-frequency resolution and low time resolution for low frequencies and high time resolution and low-frequency resolution for high frequencies [20] while STFT uses uniform time resolution for all frequencies. As its name suggests, DWT analyzes signals in terms of wavelets and more detailed explanation on DWT is presented in [21].

Time-frequency representation calculated by STFT is called by spectrogram, and its frequency scale is linear. In many cases, a full-resolution spectrogram is not needed and more abstract representation is used. Mel-spectrogram is one of the popular choices that reduces the dimensionality of the spectrogram using mel-scale. The human auditory system does not perceive pitch in a linear manner, but in approximately logarithmic above 1kHz and linear below [22], and mel-scale is a mapping function between actual frequency and human perception. Various hand-crafted audio features are based on frequency domain representation, and it is widely used as an input data for feature learning algorithm as well, which is separately explained in Section 2.2.2.

2.1.3 Cepstral Domain Representation

A frequency of the signal is important information of the audio, and it is usually obtained by applying DFT on the time domain signal as presented in the previous chapter. A human perceives fundamental frequency ($F0$) of the audio signal as 'pitch' of the sound which is an essential component of the music, and it can be measured by estimating the periodicity exist in the frequency domain signal. This 'spectrum of the spectrum' is called 'cepstrum', derived by reversing the first four letters of 'spectrum', and it is defined as the power spectrum of the logarithm of the power spectrum [23]. The frequency obtained from the spectrum is called 'quefrency', the filtering process in the cepstral do-

main is called 'liftering'. It was originally developed for analyzing the seismic echoes [24], but it is now widely used for audio processing fields such as speech processing and MIR.

A Formant is a highly important spectral characteristic which is defined as peaks of the spectral envelope. It represents vocal tract resonance in speech processing, thus it can be used for differentiating vowels [25], and similarly, it can be used for characterizing musical instruments in MIR tasks because each instrument has its unique resonance [26]. Cepstrum has clear advantage compare to spectrum in handling formant, because fundamental frequency and formant can be represented as the high quefrency and the low quefrency component.

In addition, cepstrum is for calculating MFCCs, one of the most popular audio feature that effectively captures spectral envelope related information. Unlike usual cepstrum, MFCCs are obtained by applying discrete cosine transform (DCT) on the mel-scale spectrogram to reduce the feature dimension [27]. It is possible to observe in Section 2.2.1 that most of the conventional approaches in MIR use MFCCs in the system, owing to its superior performance in various tasks.

2.2 Feature Extraction

The term 'feature extraction' used to mean hand-crafted feature only in the last decade. However, there are an increasing number of attempts for learning feature through machine learning techniques instead of rule-based deterministic approach. Although learning approach quickly substituting the conventional approach for many MIR tasks, hand-crafted features are still highly important because it can be used as an input for the feature learning algorithm, also it does not require an enormous amount of data as for feature learning. In this

section, we explain hand-crafted features and feature learning approaches for MIR tasks in detail.

2.2.1 Conventional Hand-crafted Features

Traditionally, most of the MIR tasks are based on hand-crafted features which are carefully designed with domain knowledge and signal processing techniques to extract certain target aspects from the music signal. Features are usually extracted from either raw time domain signal or frequency domain signal using DFT. For instance, a pitch of the note is extracted by fundamental frequency estimation algorithms such as harmonic product spectrum (HPS) [28] or YIN [13], performed on frequency domain and time domain, respectively.

There are a number of spectral features used across MIR tasks, and we can use them to extract higher level meaning from the raw audio data. To name a few, spectral centroid measures 'center of mass' of the spectrum, and it is reported as closely related to the timbral brightness of the sound [29]. Also, spectral flatness indicates how flat the spectrum is which is related to how much the sound is 'white-noisy', and spectral flux measures the amount of spectral change which is related to the onset of the note [30], usually used for the onset detection. One of the most popular hand-crafted features is MFCCs, a feature that contains spectral envelop-related information. It is explained in detail in Section 2.1.3. Moreover, there are a number of temporal and harmonic-related features which are usually used together with spectral features to solve MIR tasks. More information about various hand-crafted features is explained in [30].

Hand-crafted features are highly useful to measure certain acoustic characteristics, but might not be the optimal choice for identification or classification tasks. In this thesis, we focus on feature learning approaches using neural network techniques rather than using hand-crafted features, because it can adap-

tively learn a suitable feature representation according to the task and have shown superior performance across domains as mentioned in Section 1. In the following section, we introduce feature learning approaches used in the thesis, sparse feature learning and ConvNets.

2.2.2 Feature Learning Approaches

Although the research on the feature learning techniques such as neural network started in the 1940s and widely used from 1950s [31], it has not been drawn much attention by researchers until huge success of image classification work using convolutional neural network [32].

It is reported that using learned features has matched or outperformed hand-made features in a range of different modalities [33]. Recently there have been an increasing number of attempts to apply various feature learning methods to the music information retrieval field. Henaff et al. applied a sparse coding algorithm to a single frame of a constant-Q transform spectrogram for musical genre classification [34], and Schülter et al. applied restricted Boltzmann machines (RBMs) to similarity-based music classification [35]. In addition, Nam et al. applied sparse RBMs to music annotation and piano transcription [36, 37].

Despite the emergence of the feature learning approach in the MIR field, to the best of our knowledge, it has not been still widely used for musical instrument identification task. An example we found is the application of sparse coding to the cepstrum of the input signal by Yu et al. The system achieved an accuracy of 95.5% for classifying studio-recorded solo tones of 10 possible instruments [26]. Unlike the other studies listed above using individual notes, 273 solo pieces of a single instrument were evaluated for 111 seconds per audio piece on average. They also reported their classification result on a multi-source database, which was about 66%.

In particular, the case of using stacked multiple layers of neurons is called 'deep learning'. As implicated in this term, it models raw data into multiple levels of abstraction to learn higher level features. It has been reported that ConvNet, one of deep learning variants, has outperformed previous state-of-the-art approaches for various MIR tasks such as onset detection [38], automatic chord recognition [39, 40], music structure/boundary analysis [41, 42], and music genre and artist classification [43].

Deep learning approaches have shown superior performance on other signal processing fields such as speech processing and computer vision. A basic architecture of deep learning is called deep neural network (DNN), which is a feedforward network with multiple hidden layers of artificial neurons. DNN-based approaches have outperformed previous state-of-the-art methods in speech applications such as phone recognition, large-vocabulary speech recognition, multilingual speech recognition, and noise-robust speech recognition [44]. Computer vision field is probably one of the areas where deep learning approach is most extensively used. A work from Krizhevsky et al. on image classification using ConvNet, later named AlexNet, made huge performance increment compare to previous approaches in 2012 [32], and most of the state-of-the-art algorithms on ImageNet challenge such as GoogLeNet [45], VGGNet [46], and deep residual network (ResNet) [47] are based on ConvNet at the moment.

As explained above, feature learning approaches have outperformed many of previous approaches which are mostly based on hand-crafted features across domains, when appropriate input data, learning algorithms, and pre/postprocessing methods are used. Before we explain the feature learning algorithms used in this thesis in Section 2.4, we summarize the audio and feature preprocessing techniques first in the next section.

2.3 Preprocessing

As mentioned in the beginning of this chapter, preprocessing part is usually briefly mentioned or not included in MIR researches unless it is highly novel approach. It is because preprocessing methods are nearly always employed in the same manner which is clear to the community. However, preprocessing step could be critical factor for the performance and it could be a huge hurdle for researchers from the other fields. In this section, we demonstrate the preprocessing methods that are commonly used in the MIR researches.

The term 'preprocessing' indicates the process that is applied to the input data prior to feature extraction. However, scaling the obtained feature representation prior to feed them into classifier or machine learning model also usually called preprocessing. First, we are going to explain preprocessing of the input data.

A sound is usually recorded in stereo which means that there are two sound source exist, left (L) and right (R) channels. In the case of mono recording, audio signal can be directly used as a one-dimensional array, but this two channel input is normally converted to mono by taking middle channel (Mid) which is defined as $(L + R)/2$ unless the target task requires two channel input. Depending on the task, sometimes using the sound from the side (Side) only is beneficial which can be obtained by $(L - R)/2$.

Once stereo input is converted to mono, usually amplitude normalization is applied on the signal, because recordings are frequently in different scales depending on the recording environment. Unlike images, raw audio data oscillates with center at zero which means that there are both of negative and positive values. Hence, normalization is done by dividing the signal by the maximum of absolute amplitude.

For time-frequency domain representation such as spectrogram, its spectral axis is often converted to Decibel scale calculated by $20 \log$ with base-10 logarithm, because it is closer to the human ear's response [48].

As mentioned above, processing extracted features prior to feed them into classifier or machine learning model is also called preprocessing. In this case, usually obtained features are scaled to have zero mean and standard deviation at one. It can be seen as one of variations of normalization, but specifically it is called standardization, and it is done by subtracting mean and divided by its standard deviation. Usually, standardization is done for each feature which makes all the feature contribute to the model equally, not allowing the model is dominated by the feature with larger number in nature. Once the feature is standardized, it is safe to use as an input for the most of machine learning algorithms such as SVM and random forest as well as various neural network algorithms.

In this section, we introduced common preprocessing techniques used in audio processing field. Although most of them can be easily applied, preprocessing should be carefully done because it has huge influence on the result.

2.4 Feature Learning Algorithms

Recent efforts in machine learning have sought to define a method to automatically learn higher-level representations of input data [49]. Such an approach is extremely valuable, particularly when designing features by hand is challenging, and the timbre is an example where it is difficult to determine a distinctive difference in a sound spectrum by observation. In this thesis, we aim to solve instrument identification and tone detection tasks using learned features. Mainly, we use two feature learning algorithms which are sparse feature learning and

convolutional neural network. In this section, we explain these algorithms in detail.

2.4.1 Sparse Feature Learning

Many choices are available for feature learning algorithms, such as restricted Boltzmann machines [50], autoencoder [51], and sparse coding [52]. These approaches have been successfully applied to a variety of modalities, but they require extensive tuning of the hyperparameters [33].

From these approaches, we chose to employ sparse filtering as a feature learning algorithm in the proposed system because it has only one hyperparameter (the number of features to learn) and converges more quickly than other algorithms, especially when the number of input data and the feature dimension are large. This characteristic is suitable for our task because it can be easily implemented in a real-time score-following system for mobile applications without fine-parameter tuning for each device and environment. Furthermore, a short-time Fourier analysis of audio wave inherently generates a significant number of input data.

Sparse filtering first normalizes each feature to be uniformly active in total by dividing each feature by its $\ell 2$-norm throughout all examples, as follows:

$$\acute{f}_j = f_j / \left\| f_j \right\|_2 \tag{2.2}$$

where f_j represents the j^{th} feature value. In a similar manner, it then normalizes each example by dividing each example by its $\ell 2$-norm across all features, as follows:

$$\tilde{f}^{(i)} = \acute{f}^{(i)} / \left\| \acute{f}^{(i)} \right\|_2 \tag{2.3}$$

where $f^{(i)}$ represents the i^{th} example. By computing (2.2) and (2.3), now all values lie in the unit-$\ell 2$ hypersphere. This feature normalization introduces competition between features. For example, if one component of is increased, other components will decrease because of the normalization. Finally, the normalized features are optimized for sparseness using an $\ell 1$ penalty, and it can be written as

$$minimize \sum_{i=1}^{N} \left\| \tilde{f}^{(i)} \right\|_1 = \sum_{i=1}^{N} \left\| \frac{\acute{f}^{(i)}}{\left\| \acute{f}^{(i)} \right\|_2} \right\|_1 \tag{2.4}$$

for a dataset of N examples. Consequently, each example is represented by a small number of active sparse units, and each feature is active only for a small number of examples at the same time. A more detailed description of sparse filtering can be found in [33].

As an activation function, we used the soft-absolute function shown below:

$$f_j{}^{(i)} = \sqrt{\epsilon + (w_j{}^T x^{(i)})^2} \approx \left| w_j{}^T x^{(i)} \right| \tag{2.5}$$

where w is the weight matrix, x is the visible node (i.e., input data), and we set $\epsilon = 10^{-8}$. An off-the-shelf L-BFGS [53] package is used to optimize the sparse filtering objective until convergence.

The term 'feature learning' and 'deep learning' are often confused. To be precise, deep learning is feature learning with multiple layers, but not all feature learning should be deep. Going deeper or not is highly depend on algorithms and tasks, and often shallow learning (i.e., using single layer) performs better or shows matching performance. It usually happens when the target characteristic is not a very high level, thus deeper architecture does not provide an advantage. For instance, our initial work on flute fingering detection [54], not presented in this thesis, have shown that performance of the single layer sparse

feature learning matched the performance of the double layers for flute fingering detection. In this case, stacking more layer would not be a good choice in terms of computational cost. Hence, we used single layer sparse filtering in feature learning step for 'detecting fingering of overblown flute sound' in Chapter 3 as well as 'instrument identification in monophonic sound' in Chapter 4.

2.4.2 Convolutional Neural Network

ConvNet is one of neural network variants that is useful for processing data with local groups of values that are highly correlated, forming distinctive local characteristics that might appear at different parts of the array [55]. It is one of the most popular approaches recently in the image processing area such as handwritten digit recognition [56, 57, 58] for the MNIST dataset and image tagging [32, 59] for the CIFAR-10 dataset. In addition, it has been reported that it has outperformed state-of-the-art approaches for several computer vision benchmark tasks such as object detection, semantic segmentation, and category-level object recognition [44], and also for speech-recognition tasks [60].

ConvNets are usually composed of convolutional layers and pooling layers. The convolutional layer is organized with feature maps, and each unit in the feature maps is connected to the certain size of the local window in the feature maps of the previous layer through filter bank which is a set of weights [55]. Each feature map scan the input data (or output of the previous layer) and store the states at corresponding locations in the feature map which are equivalent to a convolution with a small size kernel [61]. Then, its output is passed through activation function such as hyperbolic tangent (tanh), sigmoid, or rectified linear unit (ReLU). ConvNet uses the concept of shared weight in the convolutional layer which means that the same filter bank is used for all units, and it significantly reduces computational cost.

A Pooling layer is normally inserted between convolutional layers. A common way to perform pooling is taking local maximum or mean value using a small sliding window, which makes the system more robust against local distortions and small shifts. Also, it greatly reduces the computational cost of the next layer because it is the abstraction of the previous layer. Usually, first convolutional layer learns edge-like features (e.g., horizontal, vertical, and diagonal lines) which are the smallest components to describe shapes in the input data. By stacking multiple convolutional layers and pooling layers, it is possible to extract higher level feature by a combination of lower level features from the previous layer.

The time-frequency representation of a music signal is composed of harmonics from various musical instruments and a human voice. Each musical instrument produces a unique timbre with different playing styles, and this type of spectral characteristics in music signal might appear in a different location in time and frequency. It is similar to objects in the image except the fact that sounds are more likely to be overlapped in a time-frequency representation. These hierarchical network structures of ConvNets are highly suitable for representing music audio, because music tends to present a hierarchical structure in time and different features of the music might be more salient at different time scales [62]. We utilize ConvNet for predominant instrument recognition in polyphonic music, which is presented in Chapter 5.

In this chapter, we presented an overview of MIR systems for identification tasks, and explained input data representation, feature extraction, and preprocessing methods as well as feature learning algorithms used in the thesis. From the next chapter, we explain how we applied feature learning techniques for detecting fingering of overblown flute sound, followed by experiments on the instrument identification in monophonic sound and real-world music.

Chapter 3

Detecting Fingering of Overblown Flute Sound

3.1 Introduction

Tone production on a flute involves the speed of an air jet across the embouchure and the fingering used [63]. Every pitch on the flute has its own standard fingering with an appropriate blowing pressure. However, it is possible to generate a tone that is higher in frequency than the usual tone by increasing the blowing pressure [64]; this is referred to as 'overblowing' or 'harmonic' by flutists. This implies that it is also possible to generate the same-pitched sound with different fingerings. For instance, a C6 tone can be produced using a C4 or C5 fingering as well as a C6 fingering. However, each fingering produces a sound with a slightly different timbre.

Although standard fingering generates a brighter and more stable tone than an overblown sound, alternative fingerings are frequently used to minimize mistakes in note transition or to add a special color to the tone, especially in con-

temporary repertoire [65]. On the other hand, novice players unintentionally produce overblown sounds because the fingering rules of a flute are not always consistent. In particular, most of the octave-4 and -5 fingerings are identical except for C, C#, D, and D#, and thus many beginner flute players use octave-4 fingerings for octave-5 notes.

Since the timbral differences among the same-pitched notes with distinct fingerings are not obvious, music transcription systems typically focus only on detecting the onset and pitch of a note, discarding the fingering information. However, it is valuable to figure out which fingering is used to generate a specific note, particularly for musical training purposes. Handcrafted audio features such as mel-frequency cepstral coefficients (MFCCs) are commonly used for musical timbre analysis, but there is an increasing interest in learning features from data in an unsupervised manner.

This chapter includes a method that attempts to estimate the fingering of a flute player from an audio signal based on sparse filtering (SF), which is an unsupervised feature-learning algorithm [33]. In our previous work [54], we demonstrated that the learned features could be used to detect the mistakes of beginner flute players. However, the performance of the learned features was nearly identical to that of the MFCCs, which are among the most widely used handmade features for timbre analysis. Furthermore, the evaluation was limited in that only octave-related fingerings were considered for binary classification, and the same flute was used to create all of the sound samples, thus failing to generalize for all types of overblowing.

In this chapter, we extend our previous work with a more complete evaluation by developing a fingering dataset that contains flute sound samples from various materials and makers. A major motivation for this work is to determine whether the performance of learned features shows a consistent increase

in performance over conventional features under various types of real-world situations, because learned sparse features have been shown to outperform handmade features such as MFCCs for various music information tasks such as genre classification and automatic annotation [66, 43].

To avoid classification errors caused by the blowing skill of the performer, we recorded sound samples from players with a minimum of three years' experience. In addition, we expanded the dataset to include more target pitches up to octave 6, and experimented with up to five different fingerings for each selected note. In terms of our algorithm, we changed the input for feature learning from linear-scale spectrogram and MFCCs to a mel-scale spectrogram. We also added max-pooling to make the system more robust.

The remainder of this chapter is organized as follows. We present related works in Section 3.2 and describe the spectral characteristics of an overblown tone in Section 3.3. In Section 3.4, we present the overall structure of the proposed system, including the preprocessing, feature-learning, max-pooling, and classification steps. In the following section, we explain the evaluation procedure, including the process of dataset construction, in detail. In Section 3.6, we present the results of the experiments and discussions, followed by the conclusions and directions for future work in Section 3.7.

The term 'flute' is widely used across cultures and can refer to one of the various types of woodwind instruments. In this chapter, 'flute' means a modern open-hole Boehm flute, which is the most common member of the flute family and a regular member of symphony orchestras in the West [67]. This chapter is based on the research published in EURASIP Journal on Audio, Speech, and Music Processing [68].

3.2 Existing Approaches

A study by Kereliuk et al. [69] and Verfaille et al. [70] attempted to detect overblown flute fingering using the residual noise spectrum with principal component analysis (PCA) and linear discriminant analysis (LDA). This approach uses energy measurements of multiples of the fundamental frequency submultiples F0/l where (l = 2, 3, 4, 5, 6) in the first octave and a half (i.e., between F0 and 1.5F0). This is where, in their observation, the most noticeable differences appear [70]. The spectrum energy is measured using a Hann window centered on the region of interest. In addition, the researchers use a comb-summed energy measure, which simply sums the energies of the same harmonic comb, to reduce dimensionality in the experiment. Also, the researchers recorded a flute sound from a microphone attached to the flute head joint, and only the attack segments of the notes were used in the experiment. Their proposed method allows for a detection error below 1.3% for notes with two and three possible fingerings. However, this error dramatically increased up to 14% for four and five possible fingerings. Although this system is specifically designed for detecting overblown flute sounds by measuring energy in the region of interest, from the experimental results, it is possible to say that the system does not capture all existing spectral differences between the sounds from each fingering. We plan to solve this limitation by replacing hand-designed features with sparse feature learning.

3.3 Spectral Characteristics of Overblown Tone

Timbre is often described by musicians with adjectives such as full, rich, dull, mellow, and round [71] because it is associated with the perceived feeling of the sound derived from various spectral characteristics. A timbral difference

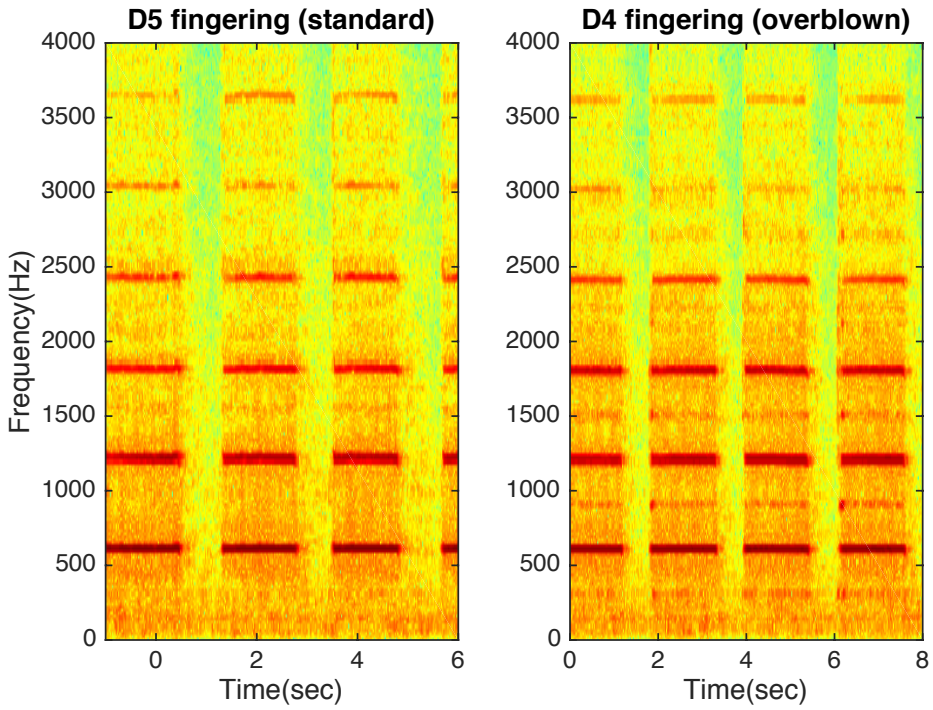

Fig. 3.1 Log spectrogram example of standard and overblown tone. Standard tone is D5 tone played with regular D5 fingering, and overblown tone is D5 tone played with D4 fingering with a sharper and stronger air jet.

between the sound generated by standard fingering and overblown fingering is actually very small; thus it is difficult for non-musicians to spot the difference. As shown in Fig. 3.1, spectrograms of a D5 tone with proper D5 fingering and D4 fingering look quite alike. However, flute experts can easily distinguish the difference. Usually, a standard tone is described as 'clearer,' and an overblown flute sound is slightly more 'airy' than the original tone. The sound of a regular flute tone is highly sine-wave-like, and only multiples of F0 are visibly strong, while the other part of the spectrum has very low energy. The 'airy' timbre of the overblown tone is mainly caused by the spectral energy existing at multiples of other than F0. It is difficult, but still possible, to observe from Fig. 3.1 that

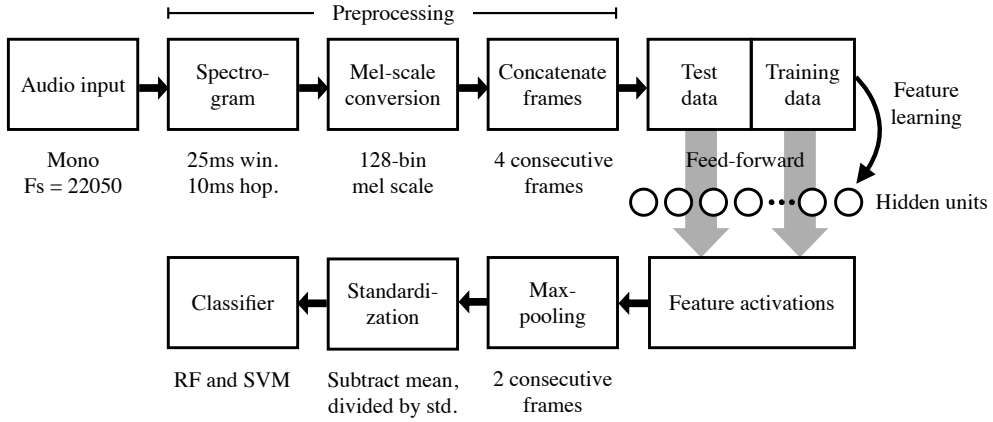

Fig. 3.2 Overall schematic of proposed flute fingering detection system. The system takes an audio waveform as an input, and four consecutive frames of a 128-bin mel-scale spectrogram are concatenated to learn timbral features. Note that features are learned only from training data. Obtained feature activations are max-pooled and standardized prior to training the classifier.

the spectrum of the overblown sound has strong peaks at multiples of F0 (587 Hz) as well as minor energy around multiples of F0 of the original fingering (293 Hz) between F0 and 2F0, and 2F0 and 3F0. Interestingly, this tendency is not clear between 3F0 and 4F0, but appears again between 4F0 and 5F0 for a D5 tone. As shown above, it is certain that an 'airy' timbre is caused by residual noise at multiples of other than F0, but it is difficult to set a concrete handmade rule for the spectral characteristics to distinguish every fingering.

3.4 System Architecture

We perform flute-fingering detection using the system architecture presented in Fig. 3.2. The details of each block in the diagram, including the parameter values, are explained in this section.

3.4.1 Preprocessing

Although sparse feature learning effectively learns the feature from the data, appropriate preprocessing is a prerequisite to obtaining a good feature as mentioned in Section 2.3. In addition, we need to decrease the dimensionality as much as possible without losing important parts of the information. We perform several steps for preprocessing, as described below.

In the first preprocessing step, the input audio is downsampled to 22,050 Hz from the original 44,100 Hz sampling rate. Since now the Nyquist frequency of the input signal is 11,025 Hz, we can get rid of unnecessary noisy information above this frequency while retaining enough numbers of harmonics for the highest note of the flute. This experiment covers frequencies up to the seventh harmonic of G6.

For time-frequency representation of the input audio, we compute a DFT for the spectrogram of the input audio with a 25 ms window and a 10 ms hop size. Then its linear frequency scale is converted to a mel scale, and the spectral magnitude is compressed by a natural logarithm. We chose 128 for the number of mel-frequency bins, following Hamel's [62] and Nam's work [37]. By using a moderate number of mel-frequency bins instead of a linear-scale DFT result, it is possible to reduce the input dimension significantly while preserving the audio content effectively enough. This is very helpful for decreasing the overall computational complexity of the system because the DFT of an audio signal generates a high-dimensional vector, and the next frame-concatenation step multiplies the number of feature dimensions.

As a final stage of preprocessing, we concatenate several consecutive frames as a single input for feature learning. This process can be seen as learning temporal information within the size of a concatenation for feature learning,

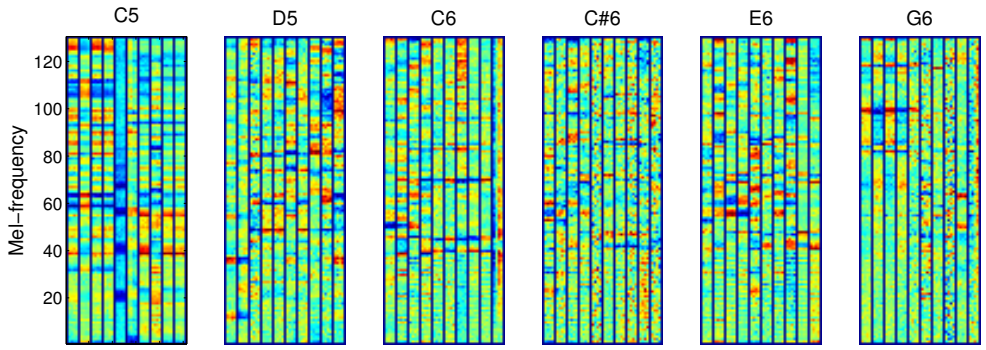

Fig. 3.3 Top 10 most active feature bases of each note. It is possible to observe that harmonic partials are distributed into bases. The bases are learned from sparse filtering with 39 hidden units.

and to make the system more robust than when we use a single frame. We concatenate four frames as a single input example. Thus the time resolution of the data in this step is now 70 ms according to the window and hop size mentioned above.

3.4.2 Feature Learning Strategy

We learn sparse features from the preprocessed data described above. Unsupervised feature learning and details of the sparse filtering algorithm is described in the Section 2.4.1. We obtained a weight matrix from training data such that the feature activation of the test data can be simply obtained by a feed-forward process. As mentioned above, sparse filtering only has one parameter to tune: the number of features to learn. To determine the effect of hidden unit size on the overall detection accuracy, we used 39, 512, and 1024 for the number of hidden units. The top 10 most active feature bases for six different notes (C5, D5, C6, C#6, E6, and G6) are shown in Fig. 3.3. It is possible to observe that harmonic partials of the mel spectrum are distributed into bases.

3.4.3 Max-pooling

Using short-term features provides high time resolution but is prone to local short-term errors. Max-pooling is the process that takes the largest value in the region of interest such that only the most responsive units are retained. We apply max-pooling over the time domain to every two non-overlapping consecutive frames. This temporal maximum value-selection process is included to add an extra robustness to the system and to reduce the computational complexity of the next step, a classifier.

It is important to note that max-pooling is a nonlinear downsampling; hence the time resolution becomes half of the original resolution when it is pooled every two frames. In the proposed system, the original time resolution of the spectrogram is 25 ms, and it becomes 70 ms after we concatenate four frames with a hop size of 10 ms. Since we are pooling over two frames, the time resolution after max-pooling is 140 ms. Increasing the size of the max-pooling region might increase the detection accuracy further, but this is an appropriate resolution for flute transcription because a typical sixteenth note with 100 beats per minute (BPM) is 150 ms, which is still greater than our frame size.

3.4.4 Classification

There are a variety of choices available for classifier. The support vector machine (SVM) and random forest (RF) are both highly popular classifiers across various applications. The underlying idea behind the SVM is to calculate a maximal margin hyperplane that performs a binary classification of the data [72]. By contrast, RF is an algorithm that uses a combination of decision trees that have a randomly selected subset of variables [73].

The performances of the SVM and RF is a highly arguable topic, and there

is significant variability in their problems and metrics [74]. We use both SVM and RF as classifiers to compare the performance. We first standardize the values by subtracting the mean, and divide them by the standard deviation prior to feeding the data into the classifier. Thus the data have their mean at zero and standard deviation at one. We use a radial basis function (RBF) kernel for SVM, and the number of trees for RF is set as 500.

3.5 Evaluation

3.5.1 Dataset Specification

We created a dataset that resembles the one in a previous study by Verfaille et al. [70]. This includes diverse overblown fingering cases such as octave-related to non-octave-related fingerings and 'few keys changing' to 'many keys changing' positions, as illustrated in Table 3.1. We recorded flute sounds from two expert performers with more than 20 years of experience and two intermediate performers with three years of experience. Each performer played six pitches with all possible fingerings as they appeared in Table 3.1. The average length of each tone was 16 s, excluding noises and silences. The length of the audio recorded from each performer was 305 s on average, and the total audio length was 1218 s. For flute tones, performers sustained the target tone for each fingering without vibrato, special articulations, or melodic phrases. We extracted 122 K frames of spectrogram in total using a 25 ms window and 10 ms hop size before frame concatenation and the max-pooling step.

The flutes used for the recording were modern Boehm flutes with B foot joints made by multiple flute makers and of different materials. Two intermediate players used silver heads with nickel body joints (Yamaha). One expert used a sterling silver flute (Powell), and another expert used a rose gold flute

(Brannen-Cooper). Note that only the first nickel body flute had a 'split E' mechanism, which facilitates the production of E6. Every flute used in the experiment was an open-hole flute without any cylindrical acrylic plugs.

Flutes made of several different materials are used for the experiment because not every flutist uses a golden flute, and the general timbre of the flute changes according to the material used in its construction. For instance, the flute made of gold is often described as having a fuller, richer, and more liquid timbre, while the silver flute is more delicate and shrill in the loud and high tones [75].

Although we used the same fingering set as [70], our data is recorded without attaching a microphone to the flute head joint. To simulate a real-world situation, audio samples were recorded using the built-in microphone of a Samsung NT900 laptop rather than by using high-end studio microphones. Flute sounds from four different performers were all recorded in different locations. Three performers played their flutes in their apartment living rooms, and one performer played the flute in a small office. The dataset is available to download from (http://marg.snu.ac.kr/DFFD), which includes the audio files with annotations.

3.5.2 Experiment Settings

A stratified eightfold cross validation was used to evaluate the performance of the proposed method. As shown in Table 3.1, our experiment is composed of six independent classification problems, and the number of classes for each note is the number of fingerings. First, we collected flute tones from every performer, and partitioned the data into eight folds that were proportionally representative of each class. Then one fold was used for a test; the remaining folds were used for training. Note that this procedure was performed frame-wise, and the order of

Table 3.1 Selected fingering set. Played note is a pitch of the note, and each pitch is played with different fingerings by altering only the blowing pressure. The evaluation is composed of six independent classification problems, and the number of classes is the number of fingerings.

Played note	Fingerings
C5	C4, C5
D5	D4, D5
C6	C4, C6, F4
C#6	C#4, C#6, F#4
E6	A4, C4, E4, E6
G6	C4, C5, E♭4, G4, G6

the frames was random. In addition, each fold includes the audio recorded from flutes of various materials and in various recording environments. The mean and standard deviation of the detection error probability P_f (i.e., inverse accuracy) was calculated eight times via repeated cross validation for each fingering of each selected pitch. We also evaluated MFCCs to compare them with our proposed feature representation. To make a fair comparison, we used the same experiment settings, such as identical frame and hop sizes, concatenating four frames as a single example, and standardizing to have a zero mean and unit variance. Thirteen-dimensional MFCCs were used with delta and double delta for a total of 39 dimensions. As mentioned previously, we evaluated the performance of sparse features with 39 hidden units for a fair comparison with MFCCs, and 512 and 1024 units are also evaluated in order to determine the effects of increased hidden unit size on the overall classification accuracy.

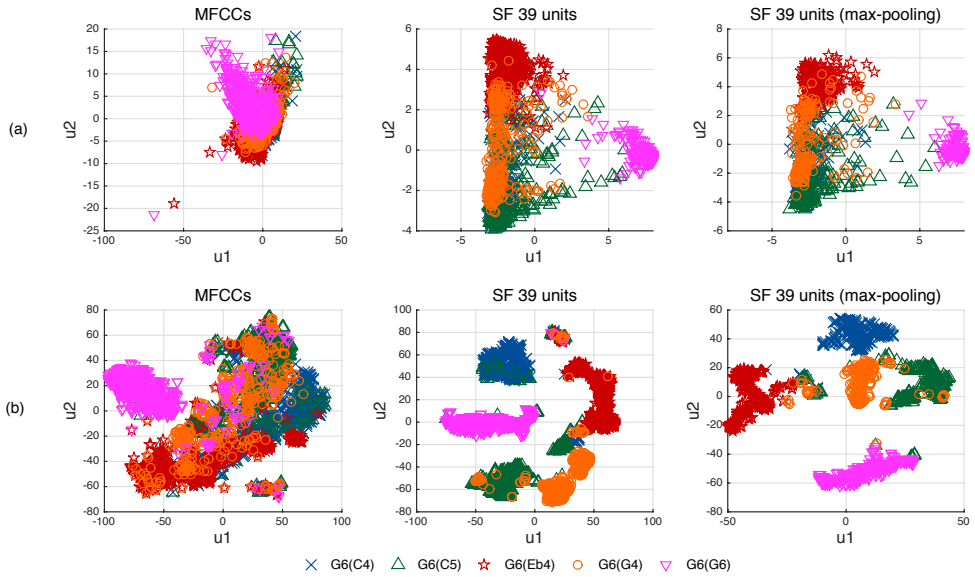

Fig. 3.4 MFCCs and sparse features with and without max-pooling projected onto a factorial map. The factorial map is built from the first and second highest eigenvalues of PCA and t-SNE (u1, u2) for visualization purposes. (a) PCA factorial map of five fingerings (C4, C5, E♭4, G4, G6) producing a G6 tone. (b) t-SNE factorial map of the same tones. It is possible to observe that sparse features are "cleaner" features for flute fingerings than the MFCCs. In addition, max-pooling removes a considerable amount of noise from the feature.

3.6 Results

The detection accuracy of the proposed flute-fingering detection method and the effect of parameters, classifiers, and max-pooling are presented in this section. We compare the performance of sparse feature representation with existing approaches such as MFCCs and the PCA/LDA method.

3.6.1 Comparison with MFCCs

In general, the proposed algorithm outperformed MFCCs for every fingering set in the same 39-dimensional setting as indicated in Table 3.2. The worst

Table 3.2 Eight fold cross-validation result of the proposed system. Mean and standard deviations of detection error ($P_f \pm \sigma^2_{Pf}$) probabilities (%) are presented for MFCCs and sparse filtering (SF) with 39 and 1024 units. Max-pooling is denoted by (Mp).

Note	Feature	SVM		Random Forest	
		$P_f \pm \sigma^2_{Pf}$	$P_f \pm \sigma^2_{Pf}$(Mp)	$P_f \pm \sigma^2_{Pf}$	$P_f \pm \sigma^2_{Pf}$(Mp)
C5	MFCCs	9.87±1.57	8.07±1.00	14.91±1.91	12.20±1.05
	SF_{39}	0.55±0.47	0.20±0.27	0.68±0.52	0.13±0.24
	SF_{1024}	0.26±0.24	0.00±0.00	0.36±0.24	0.07±0.19
D5	MFCCs	7.43±1.30	8.21±0.82	10.16±1.59	10.20±2.42
	SF_{39}	2.20±0.78	1.41±1.12	1.48±0.54	0.40±0.51
	SF_{1024}	1.67±0.72	1.18±0.68	1.00±0.36	0.39±0.38
C6	MFCCs	8.47±0.86	6.91±1.48	11.78±1.59	8.91±0.82
	SF_{39}	2.08±0.88	1.50±0.80	1.30±0.50	0.67±0.40
	SF_{1024}	0.76±0.42	0.13±0.17	0.56±0.34	0.04±0.12
C#6	MFCCs	7.01±0.62	5.47±0.90	9.31±1.01	6.02±0.98
	SF_{39}	1.39±0.72	1.07±0.82	1.26±0.52	0.56±0.33
	SF_{1024}	0.31±0.29	0.08±0.15	0.29±0.27	0.04±0.11
E6	MFCCs	6.36±0.78	4.56±0.90	7.43±0.96	4.80±1.27
	SF_{39}	2.35±0.79	1.74±0.76	2.03±0.46	1.31±0.51
	SF_{1024}	1.06±0.48	1.11±0.30	1.02±0.37	0.34±0.24
G6	MFCCs	13.80±1.14	12.77±1.28	13.76±1.12	10.63±0.96
	SF_{39}	5.71±1.30	4.49±0.76	4.62±1.26	3.22±0.58
	SF_{1024}	1.53±0.53	1.06±0.54	1.55±0.36	1.06±0.44

detection error rate of the MFCCs occurred when $P_f = 14.91 \pm 1.91\%$ for C5 with RF; the highest error rate of the proposed method occurred when

$P_f = 5.71 \pm 1.30\%$ for G6 with SVM. The proposed system does not use PCA for orthogonalization or dimension reduction. However, the first and second highest eigenvalues of PCA are computed from MFCCs and sparse features in order to visualize two-dimensional feature representation, as shown in Fig. 3.4. In addition, we also present a visualization using the t-SNE algorithm described in [76] in Fig. 3.4. This algorithm is capable of learning high-order dependencies. From this figure, it is possible to observe that MFCCs overlap significantly for all fingerings, whereas the generated sparse feature, by comparison, overlaps much less.

3.6.2 Effect of Max-pooling

Performing max-pooling on the obtained feature activation visibly improved the performance for both MFCCs and sparse features, as shown in Table 3.2. It can also be observed from Fig. 3.4 that adding max-pooling effectively removes a significant amount of feature noise. We perform max-pooling over two consecutive frames. Increasing the pooling size might increase the performance further, but we decided not to increase the pooling size in order to keep the time resolution smaller than the length of a sixteenth note at 100 BPM (150 ms).

3.6.3 Effect of the Number of Hidden Units

Although we used 39 dimensions for a fair comparison with the MFCCs, we evaluated higher values for the number of hidden units to determine their effects. Using additional hidden units means that more bases are used to describe the input signal. This effectively decreased the detection error rate. The classification performance nearly approached saturation at 256 units with a slight

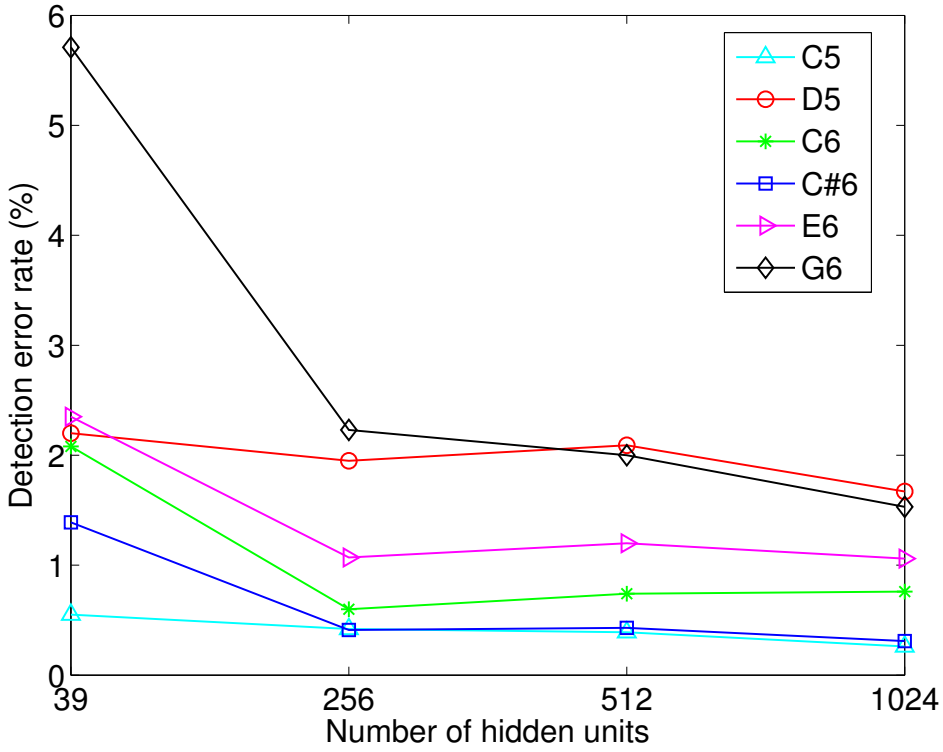

Fig. 3.5 Effect of number of hidden units. It is possible to observe that the detection error rate decreases as the number of hidden units increases. This nearly approaches saturation at 256 units, with minor improvement up to 1024 units.

improvement up to 1024 units, as shown in Fig 3.5.

3.6.4 Effect of Classifier

It is interesting to note that RF demonstrated generally matched or better performance for sparse features, and SVM exhibited superior performance for MFCCs, as shown in Fig. 3.6. For example, RF showed a lower error rate for D5, C6, C#6, E6, and G6 for SF, while SVM performed better for C5, D5, C6, C#6, and E6 for MFCCs. However, the performance gap between RF and

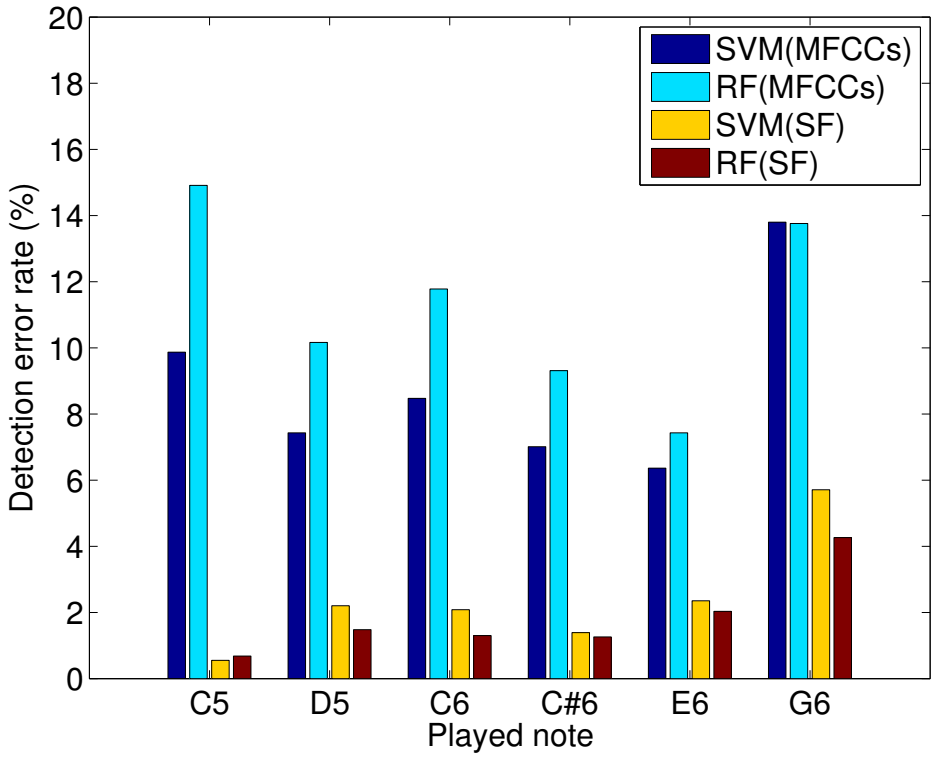

Fig. 3.6 Effect of classifier. In general, SVM and RF showed better performance for MFCCs and SF, respectively. However, using RF returned an improved result for both features as the number of possible fingerings increase.

SVM for MFCCs becomes smaller as the number of possible fingerings increases, while RF constantly returns a better performance for sparse features. We can conclude from this result that RF is more suitable for classifying sparse features, especially for multiclass tasks, and that using SVM is marginally better for MFCCs when there exist only a few possible fingerings.

3.6.5 Comparison with PCA/LDA method

A direct comparison of the PCA/LDA method from Verfaille et al. [70] with the proposed method would be inappropriate because Verfaille recorded sounds

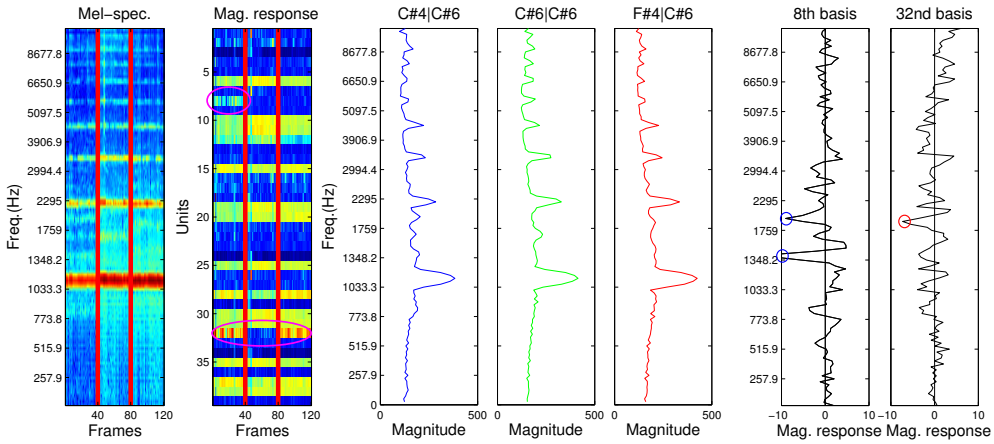

Fig. 3.7 Mel spectrogram, feature activations, mean mel spectrum, and selected basis of all possible C#6 fingerings. Mel-spectrogram and feature activations of C#4, C#6, and F#4 fingerings are separated by red vertical lines. Unique feature activations for each fingering and important spectral peaks of basis are annotated with circles.

from a microphone attached to the flute head joint and used the mean energy measured from the initial 100 ms of the attack segment only. In our proposed system, we used entire notes rather than attack segments because attack parts do not exist when the notes are played with a slur during the actual flute performance. Further, to simulate a real-world smart-device application, we used a laptop's built-in microphone at a distance to record, rather than attaching an extra microphone to the flute. However, it is meaningful to observe that the error probabilities of the proposed method were less than 5.71% for 39 sparse feature units, and less than 1.11% for 1024 units with max-pooling applied to four- and five-fingering configurations. The error rates of the PCA/LDA method were below 1.3% for two and three fingerings; however, these rates dramatically increased to 13.3% for four and five fingerings in the eightfold cross validation.

This performance improvement was a consequence of the spectral differences

between signals successfully captured by sparse feature learning, as visualized in Fig 3.7. From the magnitude response plot of C#6 in Fig 3.7, it can be observed that the eighth basis is activated for C#4 fingering; however, the eighth basis was not activated for C#6 and F#4. We can see that this eighth basis effectively captures the spectral characteristics of the C#4 fingering between the first and third octave and is particularly clear around 1348 Hz and 1905 Hz, which are annotated with blue circles. Similarly, 1855 Hz of the 32nd basis, annotated with a red circle, is significantly activated for C#4 and F#4 fingering; however, it is not significantly activated for C#6 fingering. This occurs because the sounds from C#4 and F#4 fingerings have strong energy in this region with a peak at 1905 Hz and 1806 Hz, respectively, whereas C#6 has no significant energy in this region.

The handmade feature of [70] uses energy measurements of the multiples of the fundamental frequency submultiples between the first octave and a half (i.e., F0/l where l = 2, 3, 4, 5, 6 between F0 and 1.5F0) because this is where different acoustical characteristics appear the most significant in their observation, as mentioned above. The proposed system achieved improved performance by learning spectral characteristics with sparse features and describing the sound spectrum with activations of learned bases rather than by restricting a region of interest and measuring energies at certain points.

3.7 Conclusions

We designed a flute-fingering detection system based on the sparse feature learning method. The results obtained in this study indicate that the learned sparse features delivered improved performance compared with other conventional features for flute-fingering detection, especially as the number of possible fingerings

increased. The performance gap between the MFCCs and sparse features for flute-fingering detection was not significant in our previous study [54]; however, the more complicated task with a larger dataset described in this chapter confirmed that the learned sparse features were able to capture the spectral differences among the same-pitched notes with distinct fingerings, and outperformed the conventional features such as MFCCs that focus on the spectral envelope.

The proposed method achieved a detection error rate of less than 5.71% for all cases, while the error rate of the existing PCA/LDA method dramatically increased to 13.3% for four and five fingerings. Increasing the number of units and adding max-pooling to the generated feature further improved the performance and achieved error rates of up to 1.18% for all cases. In addition, a comparison of classifier performance between SVM and RF showed that RF is generally more suitable for sparse features and is more robust against the number of classes.

The proposed system is well suited for potential use as a flute fingering detection application for education. The window/hop size for the spectrogram and the max-pooling size for the feature activation were determined while considering real-time flute transcriptions. This could be used in multiple recording environments with mobile devices because the system does not require excessive computational power or extensive parameter tuning related to the recording environment. In addition, the proposed framework can be easily applied to other instruments or timbre analysis tasks with minor changes because it does not use the deterministic rule but learns the differences from the input signal. In the next chapter, we present an instrument identification work using similar feature learning framework that used in this chapter, especially focusing on improving frame sampling, activation pooling, and frequency scaling methods.

Chapter 4

Instrument Identification in Monophonic Sound

4.1 Introduction

MIR tasks are based on audio feature extraction followed by classification as presented in 2. Traditionally, specifically designed spectral features are commonly used across MIR tasks. Such features provide highly 'intuitive' information, as each feature is designed to measure certain characteristics of the signal. However, specifically designed features usually describe the auditory characteristics of the signal in a highly condensed form and ignore many details of the input data [37].

Feature learning allows one to build systems relying less on prior knowledge of data and provides more flexibility to adapt to a given task [62]. There are a number of feature learning algorithms available, and most methods basically aim to learn a transformation matrix that maps generate a feature representation by mapping the input data to a high-dimensional sparse feature representa-

tion. It provides a simple interpretation of the input data using small elements by extracting the hidden structure of the data, thus it can learn the higher-level representations of input data automatically [49]. However, an important prerequisite to obtaining a good feature representation is appropriate data processing according to the task. For audio applications, preprocessing, frequency scaling, sampling strategy for dictionary learning, and feature summarization can be critical factors that affect the classification result.

In this chapter, we present an effective data processing method for obtaining a note-wise sparse feature representation for musical instrument identification. The term 'Instrument identification' might indicate several different research topics such as finding the predominant instrument in music [77, 78] or several instruments in a polyphonic mixture [79, 80]. In our work, we focus on classifying individual instrument notes regardless of the pitch as well as the dynamics and playing styles such as pizzicato, tremolo, vibrato, and flutter tonguing depending on the type of instrument.

This research makes several important contributions. First, we propose frame sampling methods for dictionary learning for instrument identification, which are fixed and proportional random sampling. Also, we analyze the effect of including onset frame for both of proposed sampling methods. Second, we examine standard deviation (SD) pooling for summarizing the feature activations and compare it with the commonly used max and average-pooling techniques. Third, we evaluate our system with a large-scale database with a number of tuning parameters including the analysis frame size, the dictionary size, and the type of frequency scaling as well as the different sampling and pooling methods to discover the effects of varying these variables and to find the optimal settings for the task.

The rest of this chapter is organized as follows. Section 4.2 presents the de-

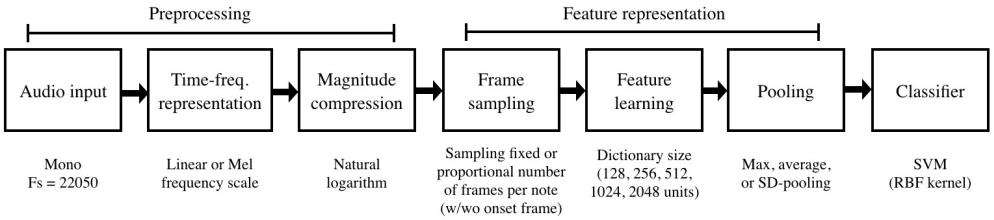

Fig. 4.1 Data processing pipeline for note-wise sparse feature representation.

tails of how we process the audio and performed feature learning. Next, Section 4.3 describes the experimental settings such as the dataset specifications and parameters. Section 4.4 reports the experimental results with the effects of sampling, pooling methods, and parameters settings. Finally, we discuss the results of various experiment settings and limitation of using isolated tone as well as a comparison to instrument identification performance of human in Section 4.5, followed by the conclusions. This chapter is based on the research published in The Journal of the Acoustical Society of America [81].

4.2 Data Processing Pipeline

We perform music instrument identification using the data processing pipeline shown in Fig 4.1. A detailed explanation of each process is presented in this section, and variable parameter settings for the experiments are separately explained in the next section.

4.2.1 Preprocessing

During the preprocessing step, the input audio is downsampled to 22,050 Hz from the original 44,100 Hz sampling rate. Because the Nyquist frequency is 11,025 Hz, it is possible to remove unnecessary noise above this frequency while retaining enough harmonic partials. Next, we compute the DFT from the down-

sampled audio to generate the time–frequency representation of the sound. Various DFT analysis frame sizes are used for the experiment to find the optimal setting, also to observe its effect on the identification result. We used analysis frame size of 512, 1024, and 2048 samples with 50% overlaps, which are 23 ms, 46 ms, and 92 ms, respectively.

It is highly common to find using mel-scale spectrogram instead of a linear scale as an input data for feature learning in other related works. It is reported that the use of a mel-scale significantly reduces the dimension while preserving enough information for tasks such as music annotation [37, 62]. In this study, both the linear-scale and mel-scale spectrograms are used for comparison because using mel-scale might not provide enough frequency resolution for instrument identification. We used 128 bin for the mel-scale spectrogram following representation learning papers on music annotation [37, 62], which is a reasonable size that sufficiently preserve the harmonic characteristics while significantly reducing the dimensionality of the input data. Finally, the spectral magnitudes are compressed with a natural logarithm prior to the feature learning process.

4.2.2 Frame Sampling Methods for Dictionary Learning

Many approaches in audio feature learning [37, 82] use randomly sampled frames for dictionary learning. By using a randomly selected frame instead of an entire training data, it is possible to reduce the computational cost and the possibility of overfitting. However, there is a relative lack of focus on 'how' and 'where' to sample the data to obtain a better feature representation. In this study, we propose more structured frame sampling methods than common random sampling, which are fixed and proportional random sampling.

For fixed-number random frame sampling, we used five and 10 randomly

selected frames per note regardless of the note length, denoted as F_5 and F_{10} throughout the paper. For proportional random sampling, we took randomly selected 1/25 and 1/13 frames from each note, denoted as P_{25} and P_{13}. Note that these denominators are selected to have approximately an equal number of frames with F_5 and F_{10}, respectively. The reason for comparing fixed and proportional sampling is to answer whether a longer audio sound needs more frames for learning or if there is a certain number of frames that is sufficient for generating a good quality dictionary.

It is reported that sound onset is important for the recognition of musical sounds [6], and some of the previous studies [3, 6] extensively use spectral features extracted during and after the onset. Hence, we further experimented with adding onset frame on the proposed fixed and proportional random sampling method in order to analyze its effect on feature learning process. The onset of each note is obtained with the spectral flux [83], which measures the positive changes in magnitude in each frequency bin. Spectral flux (SFL) is calculated from a spectrogram X as follows:

$$SFL = \sum_{k=N/2}^{N/2-1} H(|X(n,k)| - |X(n-1,k)|) \tag{4.1}$$

where n, k denote the frequency and time indices, respectively, N is the window size, and $H(x) = \frac{x+|x|}{2}$ is a half-wave rectifier function to make only positive changes matter. From the spectral flux of the signal, first we perform peak picking by finding local maxima. For music excerpts, the next step is usually to find onsets through adaptive thresholding. However, we simply pick the peak with the highest spectral flux value in this experiment because every note is separated prior to the onset detection process. More details on start/end point detection is described in following experiments section.

We choose the spectral flux as an onset detection function due to its robustness against the various onset types. Dixon reported that spectral flux providing F-measure over 0.95 in real-music excerpts for any onset types including pitched non-percussive, pitched-percussive and non-pitched percussive [84]. Its onset detection performance decreases for the complex mixture, but we limit our target to single note instrument sound classification in this work. Further, our dataset is studio recorded single tones which include less noise compare to music excerpts, hence using spectral flux would return highly reliable onset detection result. We take a single frame from the detected onset position and use it as a substitute for one frame of F_5, F_{10}, P_{25} and P_{13}. This allows us to compare different sampling methods with the same size of sampling data. These onset-included random sampling methods are denoted as F_{5_on}, F_{10_on}, P_{25_on} and P_{13_on} throughout the paper.

We learn sparse features from the data with different frame sampling methods described above. Unsupervised feature learning and details of the sparse filtering algorithm is described in the Section 2.4.1.

4.2.3 Activation Pooling

One of the most important processes of feature learning is the summarization of features over time. This process is called 'pooling,' and the goal of this summarization step is to remove unnecessary irrelevant details while preserving the task-related important information in the region of interest. The feature pooling process has a huge impact on the system performance; hence, it is important to choose an appropriate pooling region and method. Additionally, pooling significantly reduces the computational complexity for the subsequent processing steps.

Max-pooling and average-pooling are two of the most common pooling

methods across various tasks. Especially, it is reported that max-pooling is particularly well suited to the separation of features that are very sparse [85]. However, the authors also report that it might not be optimal depending on the data and features. In this study, we propose SD-pooling, which is a common statistical metric but not widely used as a pooling method for the sparse feature representation. We introduce SD-pooling to capture the dispersion of learned bases' activation because we expect that this would capture unique characteristics of each instrument. One example of using various pooling method is a study from Hamel et al. [62], which applied feature learning for music annotation. They achieved successful results by combining various pooling methods such as mean, maximum, variance and minimum while using variance alone showed comparatively worse performance than other pooling methods. SD-pooling is similar to their variance pooling except the square root, however, we found that it works marginally better than other pooling methods for the instrument identification task unlike for music annotation.

Another issue to consider for pooling is the region of interest. A generated feature map (i.e., activation of the learned bases) has a variable cardinality because the note length of each instrument and playing style is different. To identify each note, one possible method would be pooling feature vectors over fixed number of temporally local neighborhood frames and find a dominant class in each note. However, this approach can be not reliable because some instruments might share very similar spectral characteristics for long sustain part of the note such that applying the majority voting rule possibly lead to a wrong identification result.

Moreover, the fixed-size local pooling cannot fully summarize an entire note with temporal modulation that occurs in playing styles such as vibrato and flutter tonguing. Hence, we set the pooling region as the full length of the note

such that a single note of any duration and any playing style is summarized in a single frame of the sparse feature representation. Max-pooling, average-pooling, and SD-pooling can be described as below. First, let us represent the input matrix X in a feature-wise manner as

$$X = [u_1, ..., u_d]^T \in \mathbb{R}^{d \times n} \tag{4.2}$$

where $u_k = [x_{k,1}, ..., x_{k,n}]^T \in \mathbb{R}^n$ is the vector of the kth feature for all samples. Then the feature-wise max-pooling can be described as

$$X_{m.p.} = [max(u_1), ..., max(u_d)]^T \in \mathbb{R}^{d \times 1} \tag{4.3}$$

the feature-wise average pooling can be described as

$$X_{a.v.} = [avg(u_1), ..., avg(u_d)]^T \in \mathbb{R}^{d \times 1} \tag{4.4}$$

and finally the feature-wise SD-pooling can be described as

$$X_{s.d.} = [sd(u_1), ..., sd(u_d)]^T \in \mathbb{R}^{d \times 1} \tag{4.5}$$

where max, avg, and sd are maximum, average, and standard deviation of the vector.

4.2.4 Classification

There are several choices available for the classifier. We use a SVM with a RBF kernel which is a widely used classifier across a variety of scientific disciplines. SVM is basically a binary classifier and we used one-versus-one strategy to perform multi-class classification. One-versus-one strategy uses classifiers for all combination of classes, hence $k(k-1)/2$ binary SVMs are used where k is the number of classes. We make final classification decision by voting for the

winner of each classifier. Training SVM for 24 instruments took approximately 2,700 s each time on 2.5 GHz Intel Core i5 CPU with 12GB 1333MHz DDR3 RAM.

4.3 Evaluation

4.3.1 Dataset Specification

Instrument sounds are taken from the RWC musical instrument sound database [12]. The database covers the entire pitch range of each instrument at half-tone intervals with diverse playing styles such as bowing, pizzicato, vibrato, tremolo, trill, and flutter tonguing depending on the instrument type. In addition, the database provides three levels of dynamics (forte, mezzo, piano) from three different instrument manufacturers played by all different musicians for each playing style. In producing piano data, for example, the database used three pianos (Yamaha, Bösendorfer, and Steinway), for each of 88 keys, four different playing styles (normal, pedal, repeated playing, and staccato), and three dynamic levels which results in 3168 ($3 \times 88 \times 4 \times 3$) notes. These were stored in 36 ($3 \times 4 \times 3$) files as each file includes whole pitch range of the instrument with a short silence separation.

Due to this reason, we perform energy-based silence detection prior to use them in the experiment. First, we normalize the input audio, then we detect the start and end point of the note using 0.06 s energy sum sliding window with 0.02 s hop size. We slide the window from the beginning of audio, and store the frame index as a start point of the note when the energy sum inside the window is not zero. Likewise, frame index when energy sum become zero is stored as an end point of the note. We found that the database contains undesirable tiny noises such as clicks included during the recording process. To

Table 4.1 List of 24 musical instruments used for the experiment and their abbreviations.

Instrument	Abbreviation	Instrument	Abbreviation
Pianoforte	pn	Trombone	tb
Vibraphone	vp	Tuba	tu
Accordion	ac	Horn	hn
Harmonica	hm	Sopano Sax.	ss
Guitar	gt	Alto Sax.	as
Elec. guitar	eg	Tenor Sax.	ts
Elec. bass	eb	Baritone Sax.	bs
Violin	cl	Oboe	ob
Viola	vi	Bassoon	bs
Cello	ce	Clarinet	cl
Contrabass	cb	Piccolo	pc
Trumpet	tp	Flute	fl

ignore this noises, rather than using small values for threshold, we extracted start and end points first and discarded the extracted audio length shorter than 0.1 s.

We used 24 instruments from the database, as listed in Table 4.1, excluding percussion, singing voices, and some instruments with a very small number of recordings. The total number of individual notes used in the experiment was 47,517, which is approximately 1980 notes per instrument on average, and the mean length of the each audio piece was 2.96 s.

4.3.2 Experiment Settings

We computed DFTs with analysis frame sizes of 512, 1024, and 2048 samples and Hann windows with 50% overlap, which are 23 ms, 46 ms and 92 ms, re-

spectively. For the mel-spectrogram, we used 128 bin as mentioned above. We used the dictionary size as the primary variable for feature learning. We experimented with dictionary sizes of 128, 256, 512, 1024, and 2048 units to determine its influence on the performance. Because we evaluate various combinations of parameters, a linear scale, 1024 samples, P_{13}, and 512 units are defined as the default parameters for the frequency scale, DFT size, sampling method, and dictionary size, respectively.

Several related works in audio feature learning [37, 43] take multiple consecutive frames as a single input unit for an improved accuracy. In our experiment, a single frame is used as an input unit because using multiple frames for some of our parameter settings yields a very high dimensional vector requiring too expensive computation, especially for linear-frequency scale with longer DFT size. In this study, we focus more on analyzing the effect of each audio analysis parameters, which can be potentially used in conjunction with frame concatenation technique.

We also evaluated MFCCs, a widely used handcrafted feature for timbre analysis, to compare to our proposed sparse feature representation as a baseline feature. The parameter settings were the same as one of the sparse features, and 128 bins were used for the mel-scale prior to the logarithm and DCT.

Prior to feeding the pooled feature activation into the classifier, we first standardize the values by subtracting the mean and dividing by the standard deviation so that the data have a mean at zero and a standard deviation at one. For the experiment, dataset is divided into training and testing sets using five-fold stratified cross validation thus each fold is proportionally representative of each class. Training folds are used for feature learning as well as training an SVM classifier. On the other hand, testing folds are used only for the evaluation. Mean and standard deviation of the recognition accuracy are computed for

each instrument. We do not use the number of correct notes divided by the total number of notes to obtain the overall accuracy but use the mean accuracy over all instruments because every instrument has a different number of notes depending on its pitch range and the number of playing styles.

4.4 Results

We present classification result with the effects of the tuning parameters such as the analysis frame size, the dictionary size, and the type of frequency scaling as well as the different sampling and pooling methods in this section. In addition, we compare our proposed feature representation to MFCCs. As mentioned in the previous section, we use a linear scale, 1024 samples, P_{13}, and 512 units as the default settings where possible for the frequency scale, DFT size, sampling method, and dictionary size, respectively. For all experiments, we demonstrate the results with three different pooling methods: maximum, average, and standard deviation.

4.4.1 Effect of the Sampling Method

Overall, proportional sampling outperforms other sampling methods as shown in Fig. 4.2. Especially, P_{13} exhibits the best results regardless of the pooling methods. Compared to fixed-number sampling, taking frames proportional to the note length exhibits significantly improved accuracy. Regarding the number of frames, P_{13} exhibits better performance than P_{25} but with a small margin, although P_{13} uses two times the number of frames. Similarly, the performance gap between F_5 and F_{10} is minor as well. Although the mean accuracy is slightly higher when using more frames for both of fixed and proportional sampling, it is not significant as the mean accuracy fall within an error margin of each other.

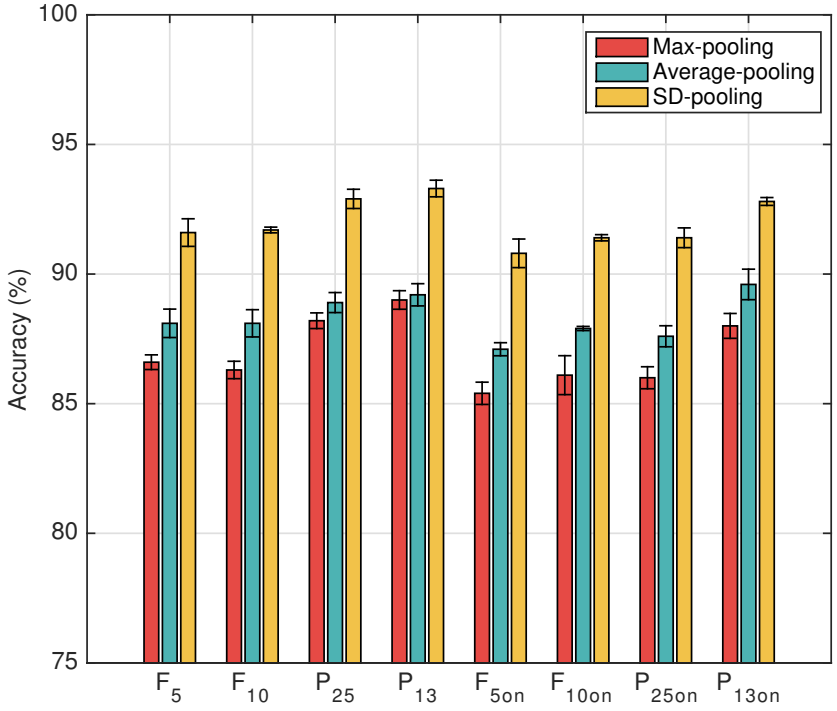

Fig. 4.2 Performance of proposed frame sampling methods with maximum, average, and SD-pooling.

On the other hand, the result shows that including the onset frame for feature learning process reduces the classification accuracy in overall. Mean accuracy of F_{5_on}, F_{10_on}, P_{25_on} and P_{13_on} are lower than F_5, F_{10}, P_{25} and P_{13} regardless of the pooling methods, except a slight improvement for the average pooling case of P_{13}.

4.4.2 Effect of the Pooling Method

Compared to max and average-pooling, SD-pooling performs constantly better for learning sparse features. With the default parameters, SD-pooling achieves

	pn	vp	ac	hm	gt	eg	eb	vl	vi	ce	cb	tp	tb	tu	hn	ss	as	ts	bs	ob	bs	cl	pc	fl
pn	98	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
vp	0	95	1	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
ac	0	0	94	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
hm	2	2	3	76	0	0	0	4	4	2	1	0	0	0	1	0	0	0	0	0	0	2	0	0
gt	0	0	0	0	98	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
eg	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
eb	1	0	0	0	1	0	97	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
vl	0	0	1	0	0	0	0	90	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
vi	0	0	1	0	1	0	0	7	87	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
ce	0	0	1	0	1	0	0	1	2	92	3	0	0	0	0	0	0	0	0	0	0	0	0	0
cb	0	0	0	0	1	0	2	0	1	4	91	0	0	0	0	0	0	0	0	0	0	0	0	0
tp	0	0	0	0	0	0	0	0	0	0	0	94	1	0	2	1	0	0	0	0	0	0	1	0
tb	0	0	0	0	0	0	0	0	0	0	0	0	96	0	3	0	0	0	0	0	0	0	0	0
tu	0	0	0	0	0	0	0	0	0	0	0	0	0	99	0	0	0	0	0	0	0	0	0	0
hn	0	1	2	0	0	0	0	2	2	1	0	3	10	0	72	1	1	1	0	0	1	1	1	0
ss	0	1	0	0	0	0	0	1	2	0	0	2	2	0	4	77	1	2	1	1	1	2	1	1
as	0	0	1	0	0	0	0	1	4	1	0	2	1	0	3	2	73	7	1	0	0	0	2	1
ts	0	0	1	0	0	0	0	0	1	2	0	1	1	0	4	4	5	74	3	1	0	0	1	1
bs	0	0	0	0	0	0	0	0	2	1	0	3	1	0	3	1	1	3	83	1	0	0	0	1
ob	0	0	2	0	0	0	0	0	1	0	0	5	0	0	0	2	1	1	0	79	0	6	0	1
bs	0	0	1	0	0	0	0	0	0	0	0	0	2	0	1	1	0	0	0	0	95	0	0	0
cl	0	0	0	0	1	0	0	1	1	1	2	0	0	0	0	0	0	0	0	0	0	91	0	1
pc	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	99	0
fl	0	1	0	1	0	0	0	2	2	1	0	2	0	0	1	2	1	0	0	0	0	0	1	85

Fig. 4.3 Confusion matrix of instrument classification. Values are mean accuracy over five-fold cross validation in percentage with a default experiment setting and max-pooling. Closely related instruments are grouped with gray shades and significant relative performance drop compare to SD-pooling (>5%) are highlighted with red boxes. X-axis and y-axis are predicted and true label, respectively.

an overall accuracy of $93.3 \pm 0.32\%$, whereas max and average-pooling achieves accuracies of $89.0 \pm 0.36\%$ and $89.2 \pm 0.43\%$, respectively. As shown in Fig. 4.3 and Fig. 4.4, SD-pooling achieves better classification accuracy than max-pooling overall. From this confusion matrix, it is possible to observe that the major performance improvement comes from the saxophone family and several wind instruments such as harmonica and oboe while classification accuracies of other instruments are slightly increased.

	pn	vp	ac	hm	gt	eg	eb	vl	vi	ce	cb	tp	tb	tu	hn	ss	as	ts	bs	ob	bs	cl	pc	fl
pn	99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
vp	0	99	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ac	0	0	95	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0
hm	2	0	2	83	0	0	0	2	3	1	2	0	0	0	3	0	0	0	0	0	0	1	0	0
gt	0	0	0	0	99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
eg	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
vl	0	0	1	0	0	0	0	94	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
vi	0	0	1	0	1	0	0	5	92	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ce	0	0	0	0	2	0	0	1	1	94	2	0	0	0	0	0	0	0	0	0	0	0	0	0
cb	0	0	0	0	1	0	2	0	1	2	93	0	0	0	0	0	0	0	0	0	0	0	0	0
tp	0	0	0	0	0	0	0	0	0	0	0	96	0	0	1	1	0	0	0	0	0	0	0	0
tb	0	0	0	0	0	0	0	0	0	0	0	0	98	0	2	0	0	0	0	0	0	0	0	0
tu	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0
hn	0	0	1	1	0	0	0	1	1	1	0	2	7	0	82	0	0	0	0	0	0	1	0	0
ss	0	0	1	0	0	0	0	0	2	0	0	2	1	0	4	80	1	1	1	0	0	2	0	2
as	0	0	1	0	0	0	0	1	2	1	0	1	0	0	1	1	85	4	1	0	0	0	1	0
ts	0	0	0	0	0	0	0	0	1	1	0	1	0	0	1	2	2	89	1	0	0	0	0	0
bs	0	0	0	0	0	0	0	0	1	1	0	1	0	0	2	1	0	2	89	0	0	0	0	0
ob	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	2	0	0	0	87	0	4	0	2
bs	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	97	0	0	0
cl	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	96	0	0
pc	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	99	0
fl	0	0	0	0	0	0	0	2	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0	93

Fig. 4.4 Confusion matrix of instrument classification. Values are mean accuracy over five-fold cross validation in percentage with a default experiment setting and SD-pooling. Closely related instruments are grouped with gray shades and significant relative performance improvements from SD-pooling (>5%) are highlighted with red boxes. X-axis and y-axis are predicted and true label, respectively.

On the other hand, SD-pooling exhibits an accuracy of $48.8 \pm 0.80\%$ for MFCCs, which is much lower than the accuracies of $68.5 \pm 0.25\%$ for max-pooling and $87.4 \pm 0.39\%$ for average-pooling. From these results, we can determine that using standard deviation is a suboptimal choice for MFCCs.

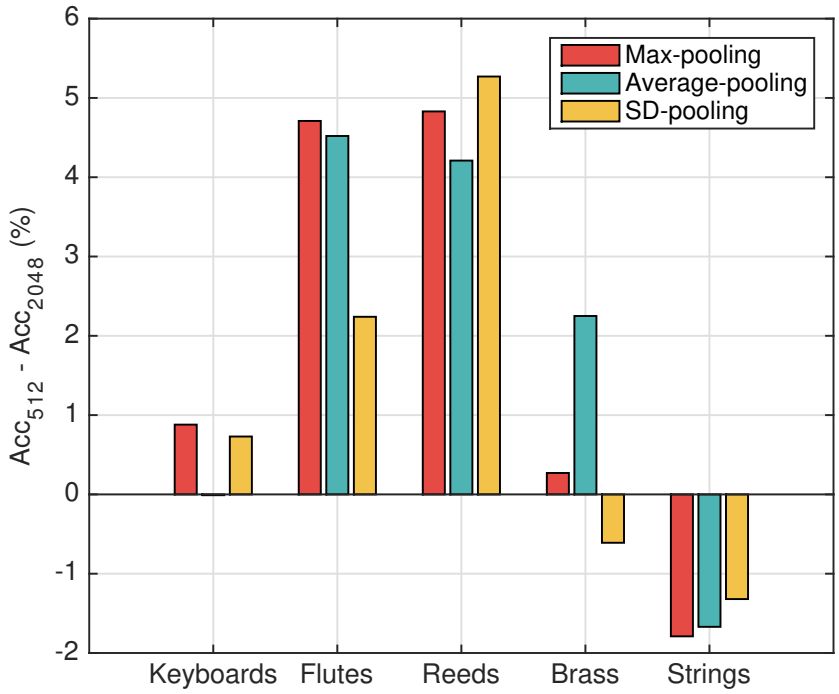

Fig. 4.5 Performance gap between DFT sizes of 512 and 2048 samples (Acc_{512} – Acc_{2048}) by instrument family.

4.4.3 Effect of the DFT size

A DFT size of 1024 samples, the default setting in our experiment, demonstrates slightly better identification accuracy compared to the use of 512 and 2048 samples. The overall performance gap between the DFT sizes was minor (within 2%); however, it is a critical factor for instrument-wise accuracy. In order to determine the effect of the DFT size, we compared the instrument-family-wise accuracy between a DFT length of 512 and 2048 samples, as shown in Fig. 4.5. We followed the instrument family from Martin and Kim23 with the exception that a keyboard is added.

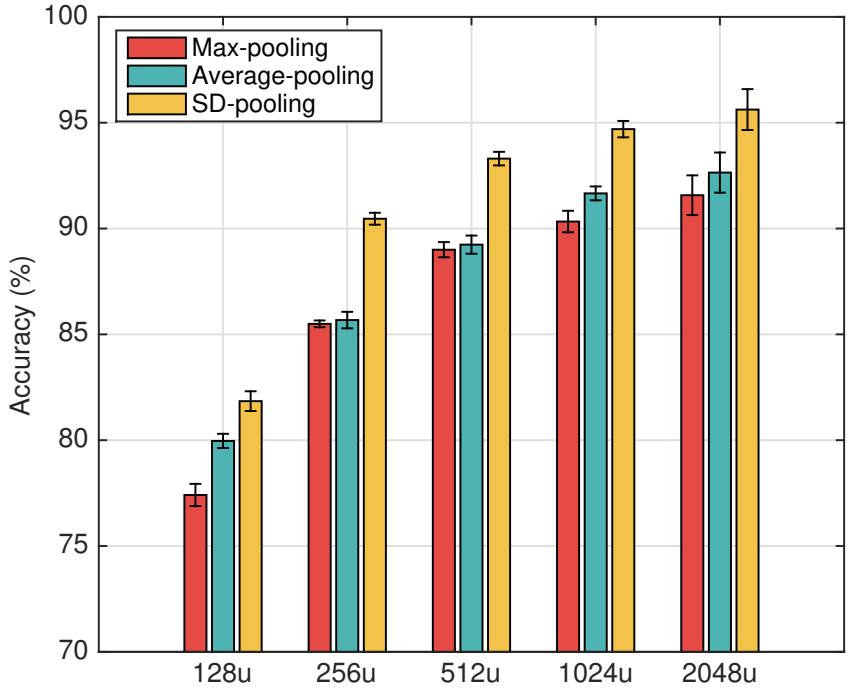

Fig. 4.6 Performance with hidden unit numbers of 128, 256, 512, 1024, and 2048 units.

From this figure, it is possible to observe that the use of a shorter DFT size generally improves the accuracy of wind instruments, and a larger DFT size is advantageous for string instruments. Our observation on the confusion matrix found that the use of 2048 samples for the DFT reduces confusion, especially among the violin, viola, and cello. Further, the use of 512 samples reduces confusion, particularly between oboe and other wind instruments.

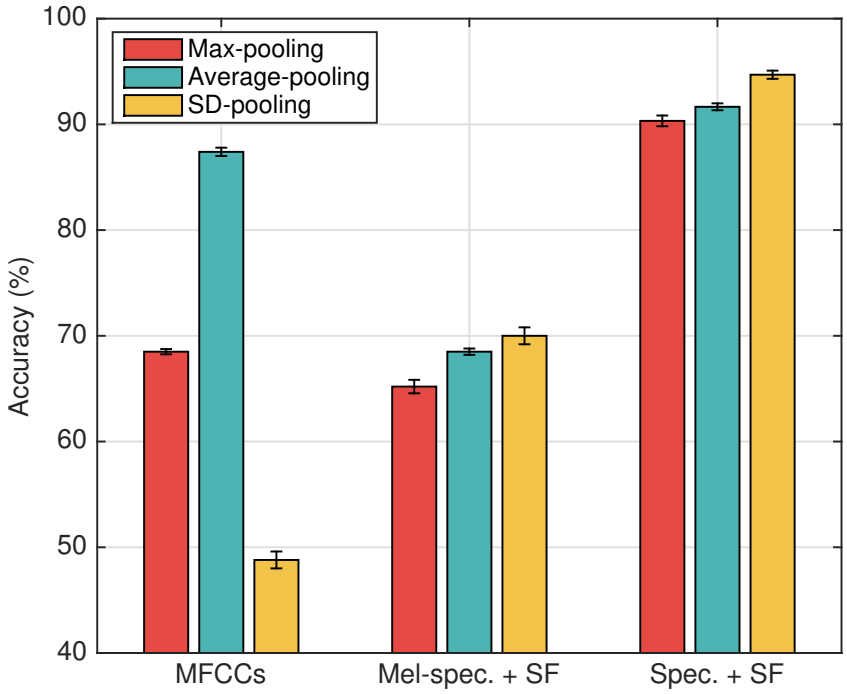

Fig. 4.7 Performance of using MFCCs, a mel-spectrogram with SF, and a linear frequency scale spectrogram with SF.

4.4.4 Effect of the Dictionary Size

The instrument identification accuracy increases as larger dictionary size is used regardless of the pooling method, as shown in Fig. 4.6. By using 2048 units, we achieved accuracies of $91.57 \pm 0.94\%$, $92.64 \pm 0.95\%$, and $95.62 \pm 0.96\%$ for max, average, and SD-pooling, respectively. Although the mean accuracy keeps increases, it nearly saturates at 1024 units and there is no significant improvement in the performance observed by increasing dictionary size up to 2048 units.

4.4.5 Effect of Frequency Scaling

Although the mel-spectrogram performs fairly well for feature learning applications in music annotation tasks [37, 62], it is visibly worse for the instrument identification task, as shown in Fig. 4.7. The use of a linear frequency scale works better, regardless of the pooling method, indicating that much information is lost by abstracting spectral energies with 128 mel-scale filter banks.

4.4.6 Comparison to MFCCs

MFCCs worked fairly well with average-pooling, exhibiting an accuracy of $87.4 \pm 0.39\%$. However, the use of a learned sparse feature from linear spectrum performs better than MFCCs, regardless of the pooling method, as shown in Fig. 4.7. With SD-pooling, the use of a learned sparse feature achieves an accuracy of $93.3 \pm 0.32\%$.

4.5 Discussion

4.5.1 Sampling and Pooling Method

The reason behind comparing two different sampling method, proportional and fixed sampling, is to examine whether there is a certain minimum amount of data to learn a good feature representation for the instrument, or it should be relative to the length of the note. Because the total number samples for F_{10} and P_{13} is nearly the same, we can say that taking more frames from a longer note and taking fewer frames from a shorter note is more effective than using a fixed number of frames regardless of note length. On the other hand, similar classification performance between F_5 and F_{10}, P_{13} and P_{13} indicates that simply using more frame of feature learning does not guarantee to produce a better feature representation.

Although the onset is musically meaningful and contains unique information, it was not helpful for generating a better feature representation, at least under the proposed data processing pipeline. From the confusion matrix, we found that this performance drop caused by onset frame is focused on few instruments. Only soprano saxophone and oboe show more than 3% of performance drop while the mean accuracies of the rest of instruments are not heavily affected by the presence of the onset frame. Most of the confusion occurred among similar types of wind instruments. Soprano saxophone is confused more with alto and tenor saxophone, and oboe is confused with clarinet and flute. This is highly likely due to the fact that feature learning for wind instrument is hindered from a transient-like spectral characteristic of the onset. For instance, the onset of soprano, alto, and tenor saxophones are similar and more significant timbral difference exists in the sustain part of the note.

Regarding pooling method, focused performance improvement of SD-pooling on wind instruments indicates that activation dispersions are important information especially for wind instruments, which can not be captured by finding the most activated basis or mean activation value. This result presents that although max-pooling is a de facto standard due to its superior performance in image processing area where neural network techniques are highly actively used, SD-pooling can be another important pooling method in the audio processing area.

On the other hand, SD-pooling showed poor performance for MFCCs unlike for learned features. This is because the amount of variation in the filter bank energies is not so meaningful for classifying the instruments, but the coefficient value itself is more powerful, contained in the widely used mean MFCCs.

4.5.2 DFT Size and the Dictionary Size

Although the overall accuracy did not heavily affect by DFT size, we could observe that using smaller size improved the accuracies of wind instruments, and a larger DFT size enhanced the performance for string instrument. A possible explanation for this behavior is that a finer temporal resolution is more important for distinguishing the timbre of wind instruments because of its rapidly time-varying characteristics, whereas a higher spectral resolution is more critical for identifying string instruments such as the violin family, which has highly similar spectral characteristics. Regarding the dictionary size, it is fairly straight forward that the performance saturates on certain dictionary size, 1024 units in this case, as a too small number of basis can not describe the spectral characteristics of the input audio.

4.5.3 Frequency Scaling and Comparison to MFCCs

The result that using linear frequency performed better than mel-frequency scale does not necessarily mean the mel-frequency conversion step removes all the important information because MFCCs with average pooling achieved better accuracy than the feature learning with mel-spectrogram as shown in Fig. 4.7. It is due to the fact that MFCCs are designed to capture spectral envelope which is less sensitive to the frequency resolution. However, it indicates that spectral resolution is critical for our proposed sparse feature learning framework which provides an interpretation of the input data using small elements by extracting the hidden structure of the data.

The result that sparse feature showing better performance than MFCCs is due to sparse feature captures the spectral difference between instruments, thus using more bases for the sparse representation leads to the better classification

accuracy as shown in Fig 4.6. In addition, the sparse feature not only learns timbral information but also learns the information about the pitch register of each instrument while MFCCs only captures information about spectral envelope. This help sparse feature to decrease the confusion further between similar instruments with different pitch range.

4.5.4 Comparison to Human Performance and Limitation of Research

Sullivan and Fujunaga reported that average recognition rate of the 27 isolated instrument tones for 88 conservatory students is 55.7% with one outstanding student recognized the instrument with 77.3% in their experiment [86]. In our experiment, we used 24 instruments which is slightly smaller variation of instruments and achieved classification result 95.62%. The performance of proposed system is marginally better than the human recognition, however, it should be pointed out that the experiment is conducted under monophonic, clean studio recording situation. Although the identification result outperformed the human performance in this case, most of the real-world music are polyphonic in various audio quality which is what humans are usually more robust at. Also, human perception makes use of much higher level information other than spectral characteristics such as note transition or guessing instruments with a genre of the music. Hence, application of this research would be limited in instrument identification for clean isolated single tone such as for music educational purpose rather than analyzing commercial music.

It is also reported that confusion mostly occurred among woodwind and brass instruments for the human [86]. It is similar tendency that our proposed system also confused more among wind instruments as shown in Fig. 4.3, but with a better overall identification performance. Using SD-pooling, we could

increase the identification performance further for wind instruments as shown in Fig. 4.4. This result indicates that identifying wind instruments is generally harder than other instruments for both of human and machine, due to its similar timbral characteristics, and using a sparse feature with SD-pooling effectively captured minor spectral differences among these instruments and their temporal dispersion which human get confused easily.

4.6 Conclusions

In this chapter, we presented a note-wise sparse feature representation for musical instrument identification. We proposed the fixed and proportional random sampling for feature learning, and extracted onset frame using spectral flux to analyze its effect on learning the process. In addition, we examined an SD-pooling method for summarizing the feature and compared it with the commonly used max and average pooling techniques.

As a result, proportional random sampling was found to be a better choice than taking fixed number of frames for feature learning. This result showed that there is no certain 'enough' amount of frame to describe notes for every instrument, but it is better to take frames with the amount relative to the length of the note. Also, the result illustrated that simply using more frames for learning does not lead to better classification accuracy. For the case of including onset frame, the overall mean accuracy slightly dropped for both of proposed sampling methods and increases confusion especially among similar wind instruments.

Regarding feature activation pooling, standard deviation was found to be a useful measure for summarizing a sparse feature for distinguishing instruments, showing better performance than taking maximum and average values. On the

other hand, the result of the same experiment on MFCCs showed that standard deviation is not a good method for summarizing filter bank energies while it works pretty well for sparse feature activations.

We also evaluated our system with a number of tuning parameters including the dictionary size, the DFT size, and the type of frequency scaling. Obviously, a larger dictionary size performed better, and the performance nearly saturated at 1024 units. Increasing dictionary size up to 2048 slightly improved the mean accuracy, but it fell within the error margin of 1024 units with a visibly larger variance of accuracy.

For the DFT size, there was no significant difference found in the overall accuracy, but the use of a shorter frame length was better for wind instruments, and a longer frame length reduced the confusion among violin families. The experimental results showed that the use of a mel-scale is a suboptimal choice for our task and our proposed sparse feature representation with SD-pooling achieved an accuracy of and $95.62 \pm 0.96\%$ with a dictionary size of 2048 units, which outperforms conventional MFCCs. We believe that the performance can be improved further by using optimal parameters, frame sampling, and feature pooling methods found in this paper in conjunction with previously discussed other feature learning techniques such as using multiple frames as a single input for dictionary learning or combining several pooling methods together.

In Chapter 3 and Chapter 4, we presented how to identify the timbral difference exist in the sound. Both pieces of research assumed the sound of instruments exists in the monophonic situation hence the application of these are limited to specific cases such as music education where clean monophonic sounds can be obtained. However, real-world music clips are usually polyphonic which means the sounds of multiple instruments are heavily overlapped. In this more complicated case, we consider that present the sound using one-dimensional

feature representation obtained by sparse feature learning would not be an optimal method. In the next chapter, we present two-dimensional feature learning approach using ConvNet to solve instrument identification problem in the polyphonic situation.

Chapter 5

Predominant Instrument Recognition in Polyphonic Music

5.1 Introduction

As explained in the Section 1.2, most of the previous works focused on the identification of the instrument sounds in clean solo tones or phrases. More recent research studies on polyphonic sounds are closer to the real-world situation, but artificially produced polyphonic music is still far from professionally produced music. Real-world music has many other factors that affect the recognition performance. For instance, it might have a highly different timbre, depending on the genre and style of the performance. In addition, an audio file might differ in quality to a great extent, depending on the recording and production environments.

In this chapter, we investigate a method for predominant instrument recognition in professionally produced Western music recordings. We utilize ConvNets to learn the spectral characteristics of the music recordings with 11 mu-

sical instruments and perform instrument identification on polyphonic music excerpts. The major contributions of the work presented in this paper are as follows.

1. We present the ConvNet architecture for predominant musical instrument identification where the training data are single labeled and the target data are multi-labeled with an unknown number of classes existing in the data.

2. We introduce a normalized mean and max-pooling method for aggregation of multiple outputs from short-time sliding windows to find the predominant instruments in a music excerpt with variable length, where the conventional method of majority vote often fails.

3. We conduct an extensive experiment with various parameters to find an optimal setting. Our analysis on the experiment found that the onset type of the instrument is a critical factor for the identification threshold.

The remainder of this chapter is organized as follows. In Section 5.2, we introduce emerging deep neural network techniques in the MIR field. Next, the system architecture section includes audio preprocessing, the proposed network architecture with detailed training configuration, and an explanation of various activation functions used for the experiment. Section 5.4, the evaluation section, contains information about the dataset, testing configuration including aggregation strategy, and our evaluation scheme. Then, we illustrate the performance of the proposed ConvNet in Section 5.5, the results section, with an analysis of the effects of analysis window size, aggregation strategy, activation function, and identification threshold. We also present an analysis on

instrument-wise analysis and single predominant instrument identification as well as a qualitative analysis based on the visualization of the ConvNet's intermediate outputs to understand how the network captured the pattern from the input data. Finally, we conclude the paper in Section 5.6. This chapter is based on the research published in IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP) [87].

5.2 Proliferation of Deep Neural Networks in Music Information Retrieval

The ability of traditional machine learning approaches was limited in terms of processing input data in their raw form. Hence, usually the input for the learning system, typically a classifier, has to be a hand-crafted feature representation, which requires extensive domain knowledge and a careful engineering process. However, it is getting more common to design the system to automatically discover the higher-level representation from the raw data by stacking several layers of nonlinear modules, which is called deep learning [55]. Recently, deep learning techniques have been widely used across a number of domains owing to their superior performance. A basic architecture of deep learning is called deep neural network (DNN), which is a feedforward network with multiple hidden layers of artificial neurons. DNN-based approaches have outperformed previous state-of-the-art methods in speech applications such as phone recognition, large-vocabulary speech recognition, multi-lingual speech recognition, and noise-robust speech recognition [44].

There are many variants and modified architectures of deep learning, depending on the target task. Especially, recurrent neural networks (RNNs) and ConvNets have recently shown remarkable results for various multimedia information retrieval tasks. RNNs are highly powerful approaches for sequential

inputs as their recurrent architecture enables their hidden units to implicitly maintain the information about the past elements of the sequence. Since languages natively contain sequential information, it is widely applied to handle text characters or spoken language. It has been reported that RNNs have shown a successful result on language modeling [88] and spoken language understanding [89, 90].

ConvNet is one of neural network variants that exploits local characteristics of the input data, which is explained in Section 2.4.2. It is widely used across computer vision fields such as image recognition and video analysis, also applied to various audio processing tasks such as speech recognition. Although ConvNets have been a more commonly used technique in image processing, there are an increasing number of attempts to apply ConvNets for music signal. It has been reported that ConvNet has outperformed previous state-of-the-art approaches for various MIR tasks such as onset detection [38], automatic chord recognition [39, 40], and music structure/boundary analysis [41, 42].

An attempt to apply ConvNets for musical instrument identification can be found in the recent report from Park et al. [91] and Li et al. [92], although it is still an ongoing work and is not a predominant instrument recognition method; hence, there are no other instruments but only target instrument sounds exist. Our research differs from [91] because we deal with polyphonic music, while their work is based on the studio recording of single tones. In addition, our research also differs from [92] because we use single-label data for training and estimate multi-label data, while they used multi-label data from the training phase. Moreover, they focused on an end-to-end approach, which is promising in that using raw audio signals makes the system rely less on domain knowledge and preprocessing, but usually it shows a slightly lower performance than using spectral input such as mel-spectrogram in recent papers [14, 15].

5.3 System Architecture

5.3.1 Audio Preprocessing

The convolutional neural network is one of the representation learning methods that allow a machine to be fed with raw data and to automatically discover the representations needed for classification or detection [55]. However, appropriate preprocessing of input data is still an important issue to improve the performance of the system.

In the first preprocessing step, the stereo input audio is converted to mono by taking the mean of the left and right channels, and then it is downsampled to 22,050 Hz from the original 44,100 Hz of sampling frequency. This allows us to use frequencies up to 11,025 Hz, the Nyquist frequency, and it is sufficient to cover most of the harmonics generated by musical instruments while removing noises possibly included in the frequencies above this range. Moreover, all audios are normalized by dividing the time-domain signal with its maximum value. Then, this downsampled time-domain waveform is converted to a time-frequency representation using short-time Fourier transform (STFT) with 1024 samples for the window size (approx. 46 ms) and 512 samples of the hop size (approx. 23 ms).

Next, the linear frequency scale-obtained spectrogram is converted to a mel-scale. We use 128 for the number of mel-frequency bins, following the representation learning papers on music annotation by Nam et al. [37] and Hamel et al. [62], which is a reasonable setting that sufficiently preserves the harmonic characteristics of the music while greatly reducing the dimensionality of the input data. Finally, the magnitude of the obtained mel-frequency spectrogram is compressed with a natural logarithm.

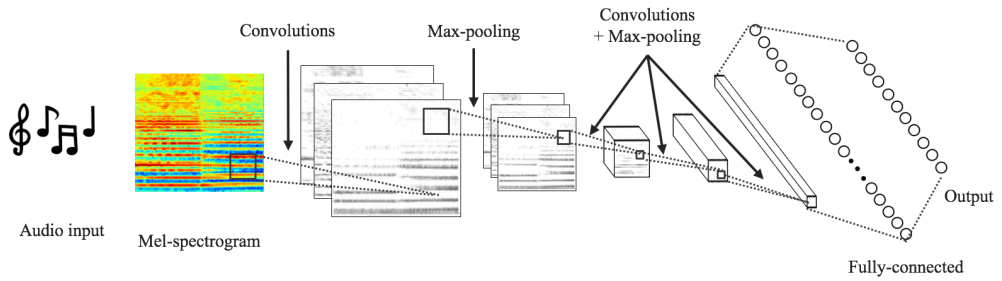

Fig. 5.1 Schematic of the proposed ConvNet containing 4 times repeated double convolution layers followed by max-pooling. The last max-pooling layer performs global max-pooling, then it is fed to a fully connected layer followed by 11 sigmoid outputs.

5.3.2 Network Architecture

ConvNets can be seen as a combination of feature extractor and the classifier. Our ConvNet architecture generally follows a popular AlexNet [32] and VGGNet [46] structure, which contains very deep architecture using repeated several convolution layers followed by max-pooling, as shown in Fig. 5.1. This method of using smaller receptive window size and smaller stride for ConvNet is becoming highly common especially in the computer vision field such as in the study from Zeiler and Fergus [93] and Sermanet et al. [94], which has shown superior performance in ILSVRC-2013.

Although the general architecture style is similar to that of other successful ConvNets in the image processing area, the proposed ConvNet is designed according to our input data. We use filters with a very small 3×3 receptive field, with a fixed stride size of 1, and spatial abstraction is done by max-pooling with a size of 3×3 and a stride size of 1.

In Table 5.1, we illustrate the detailed ConvNet architecture with the input size in each layer with parameter values except the zero-padding process. The input for each convolution layer is zero-padded with 1×1 to preserve the spa-

Table 5.1 Proposed ConvNet Structure. The Input Size Demonstrated in This Table Is for an Analysis Window Size of 1 Second (Number of filters × time × frequency). The Activation Function Is Followed by Each Convolutional Layer and a Fully Connected Layer. The Input of Each Convolution Layer Is Zero-Padded with 1 × 1, But Is Not Shown for Brevity.

Input size	Description
1 × 43 × 128	mel-spectrogram
32 × 45 × 130	3 × 3 convolution, 32 filters
32 × 47 × 132	3 × 3 convolution, 32 filters
32 × 15 × 44	3 × 3 max-pooling
32 × 15 × 44	dropout (0.25)
64 × 17 × 46	3 × 3 convolution, 64 filters
64 × 19 × 48	3 × 3 convolution, 64 filters
64 × 6 × 16	3 × 3 max-pooling
64 × 6 × 16	dropout (0.25)
128 × 8 × 18	3 × 3 convolution, 128 filters
128 × 10 × 20	3 × 3 convolution, 128 filters
128 × 3 × 6	3 × 3 max-pooling
128 × 3 × 6	dropout (0.25)
256 × 5 × 8	3 × 3 convolution, 256 filters
256 × 7 × 10	3 × 3 convolution, 256 filters
256 × 1 × 1	global max-pooling
1024	flattened and fully connected
1024	dropout (0.50)
11	sigmoid

tial resolution regardless of input window size, and we increase the number of channels for the convolution layer by a factor of 2 after every two convolution layers, starting from 32 up to 256.

In the last max-pooling layer after the eight convolutional layers, we perform

global max-pooling followed by one fully connected layer. Recently, it has been reported that the use of global average pooling without a fully connected layer before a classifier layer is less prone to overfitting and shows better performance for image processing datasets such as CIFAR-10 and MNIST [95]. However, our empirical experiment found that global average pooling slightly decreases the performance and that global max-pooling followed by a fully connected layer works better for our task.

Finally, the last classifier layer is the sigmoid layer. It is common to use a softmax layer when there is only one target label, but our system must be able to handle multiple instruments present at the same time, and, thus, a sigmoid output is used.

5.3.3 Training Configuration

The training was done by optimizing the categorical cross-entropy between predictions and targets. We used Adam [96] as an optimizer with a learning rate of 0.001, and the mini-batch size was set to 128. To accelerate the learning process with parallelization, we used a GTX 970 GPU, which has 4GB of memory.

The training was regularized using dropout with a rate of 0.25 after each max-pooling layer. Dropout is a technique that prevents the overfitting of units to the training data by randomly dropping some units from the neural network during the training phase [97]. Furthermore, we added dropout after a fully connected layer as well with a rate of 0.5 since a fully connected layer easily suffers from overfitting.

In addition, we conducted an experiment with various time resolutions to find the optimal analysis size. As our training data were a fixed 3-s audio, we performed the training with 3.0, 1.5, 1.0, and 0.5 s by dividing the training audio and used the same label for each divided chunk. The audio was divided

without overlap for training as it affects the validation loss used for the early stopping. Fifteen percent of the training data were randomly selected and used as a validation set, and the selection was done in an audio-clip scale to ensure that these are not coming from the same recording. The training process was stopped when the validation loss did not decrease for more than three epochs.

The initialization of the network weights is another important issue as it can lead to an unstable learning process, especially for a very deep network. We used a uniform distribution with zero biases for both convolutional and fully connected layers following Glorot and Bengio [98].

5.3.4 Activation Function

The activation function is followed by each convolutional layer and fully connected layer. In this section, we introduce several activation functions used in the experiment for the comparison. The traditional way to model the activation of a neuron is by tanh or sigmoid function. However, non-saturating nonlinearities such as the ReLU allow much faster learning than these saturating nonlinearities, particularly for models that are trained on large datasets [32]. Moreover, a number of works have shown that the performance of ReLU is better than that of sigmoid and tanh activation [99]. Thus, most of the modern studies on ConvNets use ReLU to model the output of the neurons [92, 46, 93, 94].

ReLU was first introduced by Nair and Hinton in their work on restricted Boltzmann machines [100]. The ReLU activation function is defined as

$$y_i = max(0, z_i) \tag{5.1}$$

where z_i is the input of the ith channel. ReLU simply suppresses the whole

negative part to zero while retaining the positive part. Recently, there have been several modified versions of ReLU introduced to improve the performance further. First, leaky-ReLU (LReLU), introduced by Mass et al. [101], compresses the negative part rather than make it all zero, which might cause some initially inactive units to remain inactive. It is defined as

$$y_i = \begin{cases} z_i & z_i \geq 0 \\ \alpha z_i & z_i < 0 \end{cases} \tag{5.2}$$

where α is a parameter between 0 and 1 to give a small gradient in the negative part. Second, parametric ReLU (PReLU), introduced by He et al. [102], is basically similar to LReLU in that it compresses the negative part. However, PReLU automatically learns the parameter for the negative gradient, unlike LReLU. It is defined as

$$y_i = \begin{cases} z_i & z_i \geq 0 \\ \alpha_i z_i & z_i < 0 \end{cases} \tag{5.3}$$

where α_i is the learned parameters for the ith channel.

The choice of activation function considerably influences the identification performance. It is difficult to say which specific activation function always performs the best because it highly depends on the parameter setting and the input data. For instance, an empirical evaluation of the ConvNet activation functions from Xu et al. [103] reported that the performance of LReLU is better than those of ReLU and PReLU, but sometimes it is worse than that of basic ReLU, depending on the dataset and the value for α. Moreover, most of the works regarding activation function are on the image classification task, not on the audio processing domain.

Hence, we empirically evaluated several activation functions explained above such as tanh, ReLU, LReLU, and PReLU to find the most suitable activation

Table 5.2 List of Musical Instruments Used in the Experiment with Their Abbreviations, and the Number of Labels of the Training and Testing Audio.

Instruments	Abbreviations	Training (n)	Testing (n)
Cello	cel	388	111
Clarinet	cla	505	62
Flute	flu	451	163
Acoustic guitar	acg	637	535
Electric guitar	elg	760	942
Organ	org	682	361
Piano	pia	721	995
Saxophone	sax	626	326
Trumpet	tru	577	167
Violin	vio	580	211
Voice	voi	778	1044

function for our task. For LReLU, very leaky ReLU ($\alpha = 0.33$) and normal leaky ReLU ($\alpha = 0.01$) were used, because it has been reported that the performance of LReLU considerably differs depending on the value and that very leaky ReLU works better [103].

5.4 Evaluation

5.4.1 IRMAS Dataset

The IRMAS dataset includes musical audio excerpts with annotations of the predominant instruments present and is intended to be used for the automatic identification of the predominant instruments in the music. This dataset was used in the paper on predominant instrument classification by Bosch et al. [77] and includes music from various decades from the past century, hence differing in audio quality to a great extent. In addition, the dataset covers a wide

variability in musical instrument types, articulations, recording and production styles, and performers.

The dataset is divided into training and testing data, and all audio files are in 16-bit stereo wave with 44,100 Hz of sampling rate. The training data consisted of 6705 audio files with excerpts of 3 s from more than 2000 distinct recordings. Two subjects were paid to obtain the data for 11 pitched instruments from selected music tracks as shown in Table 5.2 with the objective of extracting music excerpts that contain a continuous presence of a single predominant instrument.

On the other hand, the testing data consisted of 2874 audio files with lengths between 5 s and 20 s, and no tracks from the training data were included. Unlike the training data, the testing data contained one or more predominant target instruments. Hence, the total number of training labels was identical to the number of audio files, but the number of testing labels was more than the number of testing audio files as the latter are multi-label. For both the training and the testing dataset, other musical instruments such as percussion and bass were not included in the annotation even if they exist in the music excerpts.

Although the IRMAS dataset provides separate testing data, we divided them into halves to use one as a pure test set, and another one as a development set for parameter tuning. No tracks from the development set were included in the test set, and all the experiments to find optimal parameter setting were conducted with the development set.

5.4.2 Testing Configuration

In the training phase, we used a fixed length window because the input data for ConvNet should be in a specific fixed shape. However, our testing audios had variable lengths between 5 s and 20 s, which were much longer than those of the

training audio. Developing a system that can handle variable length of input data is valuable because music in real life varies in its length. We performed short-time analysis using overlapping windows to obtain local instrument information in the audio excerpts. We used the same window size from the training phase to analyze testing audios and the analysis hop size was set to half of the window size.

Because an annotation exists per audio clip, we observed multiple sigmoid outputs and aggregated them to make a clip-wise decision. First, we took an average of the sigmoid outputs class-wise (i.e., instrument-wise) over the whole input audio clip. Then, obtained values were normalized by dividing them by the maximum value among classes such that the values were scaled to be placed between zero and one. This method is based on the assumption that humans perceive the "predominant" instrument in a relatively scaled sense such that the strongest instrument is always detected and the existence of other instruments is judged by their relative strength compared to the most activate instrument.

In addition, we also conducted an experiment with max-pooling in the aggregation process. This method is based on the assumption that averaging sigmoid output might suppress the activation of instruments which appear only sporadically, but temporally local max-pooling could reduce this effect. We used sliding window to perform class-wise max-pooling prior to averaging the sigmoid outputs.

Majority vote, one of the most common choices for a number of classification tasks, is not used in our system. Majority vote first predicts the classes for each analysis frame and the one with more vote wins. However, using this method for our task would result in disregarding accompaniment instruments, piano for example, because a music signal is composed of various musical instruments and usually the sounds are overlapped in time domain, and a presence of ac-

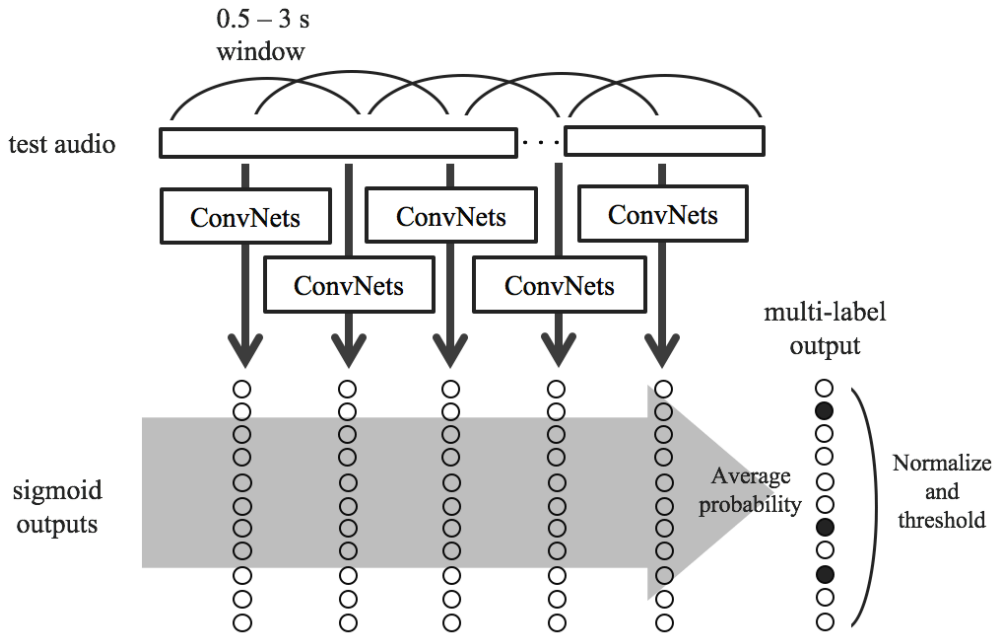

Fig. 5.2 Schematic of obtaining a multi-label output from a test audio signal. Input audio was analyzed with sliding window, and these multiple sigmoid outputs were aggregated by averaging followed by normalization to estimate the predominant instrument in an audio excerpt level.

companiments are usually much weaker than that of voice or lead instruments.

As our target is to identify an arbitrary number of predominant instruments in testing data, instruments with aggregated value over the threshold were all considered as predominant instruments. Using a higher value for the identification threshold will lead to better precision, but it will obviously decrease the recall. On the other hand, a lower threshold will increase the recall, but will lower the precision. Hence, we tried a range of values for the threshold to find the optimal value for the $F1$ measure, which is explained in the next Performance Evaluation section.

We used values between 0.2 and 0.8 as an identification threshold θ to find the optimal setting. This threshold range was empirically chosen but set to be

a wide enough range to find the best performance (i.e., highest $F1$ measure). The schematic of this aggregation process is illustrated in Figure 5.2.

5.4.3 Performance Evaluation

Following the evaluation method widely used in the instrument recognition task, we computed the precision and recall, which are defined as:

$$P_l = \frac{tp_l}{tp_l + fp_l} \tag{5.4}$$

$$R_l = \frac{tp_l}{tp_l + fn_l} \tag{5.5}$$

where tp_l is true positive, fp_l is false positive, and fn_l is false negative for each of the labels l in L.

Using more strict parameter setting will lead to a better precision with decreased the recall. On the contrary, more loose parameter setting will result in a better recall with lower precision. Hence, we used the $F1$ measure to calculate the overall performance of the system, which is the harmonic mean of precision:

$$F1_l = \frac{2P_l R_l}{P_l + R_l} \tag{5.6}$$

Since the number of annotations for each class (i.e., 11 musical instruments) was not equal, we computed the precision, recall, and $F1$ measure for both the micro and the macro averages. For the micro averages, we calculated the metrics globally regardless of classes, thus giving more weight to the instrument with a higher number of appearances. Micro precision and recall are defined as:

$$P_{micro} = \frac{\sum_{l=1}^{L} tp_l}{\sum_{l=1}^{L} tp_l + fp_l} \tag{5.7}$$

$$R_{micro} = \frac{\sum_{l=1}^{L} tp_l}{\sum_{l=1}^{L} tp_l + fn_l} \tag{5.8}$$

On the other hand, we calculated the metrics for each label and found their unweighted average for the macro averages; hence, it is not related to the number of instances, but represents the overall performance over all classes. Macro precision and recall are defined as:

$$P_{macro} = \frac{1}{|L|} \sum_{l=1}^{L} P_l \tag{5.9}$$

$$R_{macro} = \frac{1}{|L|} \sum_{l=1}^{L} R_l \tag{5.10}$$

Finally, micro and macro $F1$ measure can be written as:

$$F1_{micro} = \frac{2P_{micro}R_{micro}}{P_{micro} + R_{micro}} \tag{5.11}$$

$$F1_{macro} = \frac{2P_{macro}R_{macro}}{P_{macro} + R_{macro}} \tag{5.12}$$

We repeated each experiment three times and the mean value of them are demonstrated.

5.5 Results

Our experiment is carried out as follows. First, we find optimal analysis resolution by varying the analysis window size. Then we observe the effect of adding max-pooling and on the aggregation process, and find the optimal value for the identification threshold. Next, we try various activation functions to observe its effect on the result. Finally, we compare our result to existing algorithms using test set with optimal parameter settings found through the experiment.

Table 5.3 Experiment Variables for the Activation Function, Size of the Analysis Window, Aggregation Strategy, and Identification Threshold. Default settings are indicated in bold.

Variables	
analysis win. size	0.5 s, **1.0 s**, 1.5 s, 3.0 s
θ (threshold)	0.20 to 0.80 (step size 0.05, default **0.55**)
Activation func.	tanh, **ReLU**, PReLU, LReLU (0.01), LReLU (0.33)
Agg. strategy	**without max-pooling**, with max-pooling

We used ReLU for the activation function, 1 s for the analysis window, aggregation without max-pooling, and 0.55 for the identification threshold as default settings of the experiment where possible. The experiment variables are listed in Table 5.3. Effect of each variable is separately demonstrated in following sections, while identification threshold is discussed together with analysis window size and max-pooling as the optimal value of it is closely related with them.

5.5.1 Effect of Analysis Window Size

We conducted an experiment with diverse analysis window sizes such as 3.0, 1.5, 1.0, and 0.5 s to find the optimal analysis resolution. Figure 5.3 shows the micro and macro $F1$ measure with various analysis frame sizes according to identification threshold. Although the length of the original training data is 3.0 s, it can be observed that using full 3.0 s as a window size clearly performed poorer than smaller windows regardless of identification threshold. This result suggests that increasing the number of example by dividing training data into smaller chunks is beneficial, also it provides too coarse temporal resolution.

However, shortening the analysis frame down to 0.5 s decreased the over-

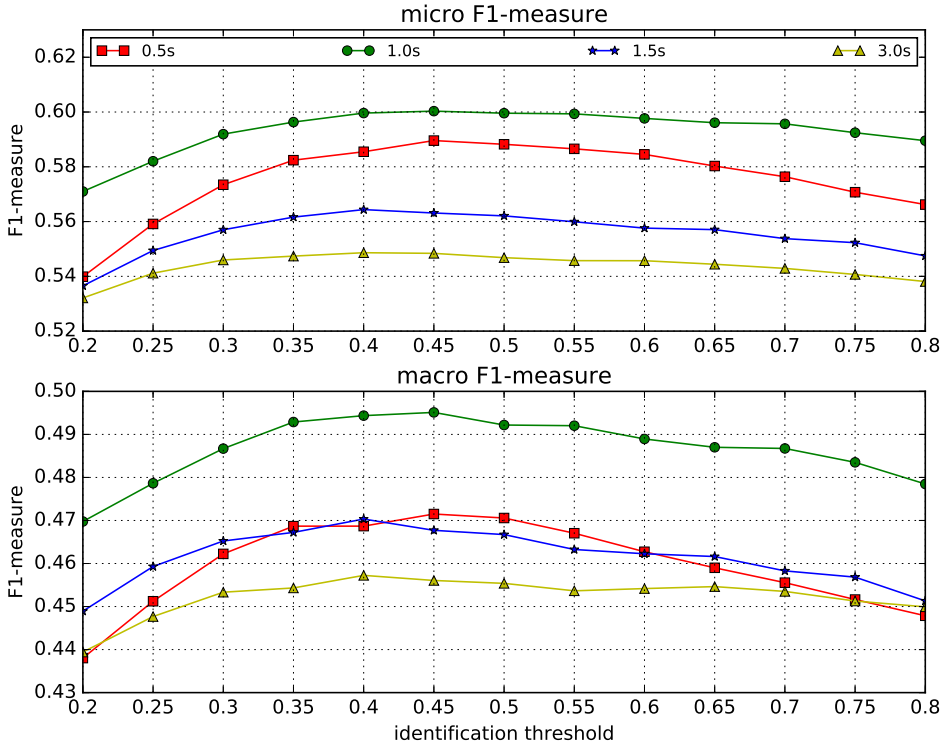

Fig. 5.3 Micro and macro $F1$ measure of an analysis window size of 0.5, 1.0, 1.5, and 3.0 s according to the identification threshold.

all performance again. Using a shorter analysis frame helped to increase the temporal resolution, but 0.5 s was found to be too short a window size for identifying the instrument. From this result, it can be seen that 1.0 s is the optimal analysis window size for our task regardless of the identification threshold.

5.5.2 Effect of Max-pooling and Model Ensemble

We conducted an experiment with adding max-pooling on the aggregation process to prevent the output averaging step suppresses the activation of instruments that appear sporadically. This class-wise max-pooling is performed prior to output averaging and normalization, by simply taking maximum value for

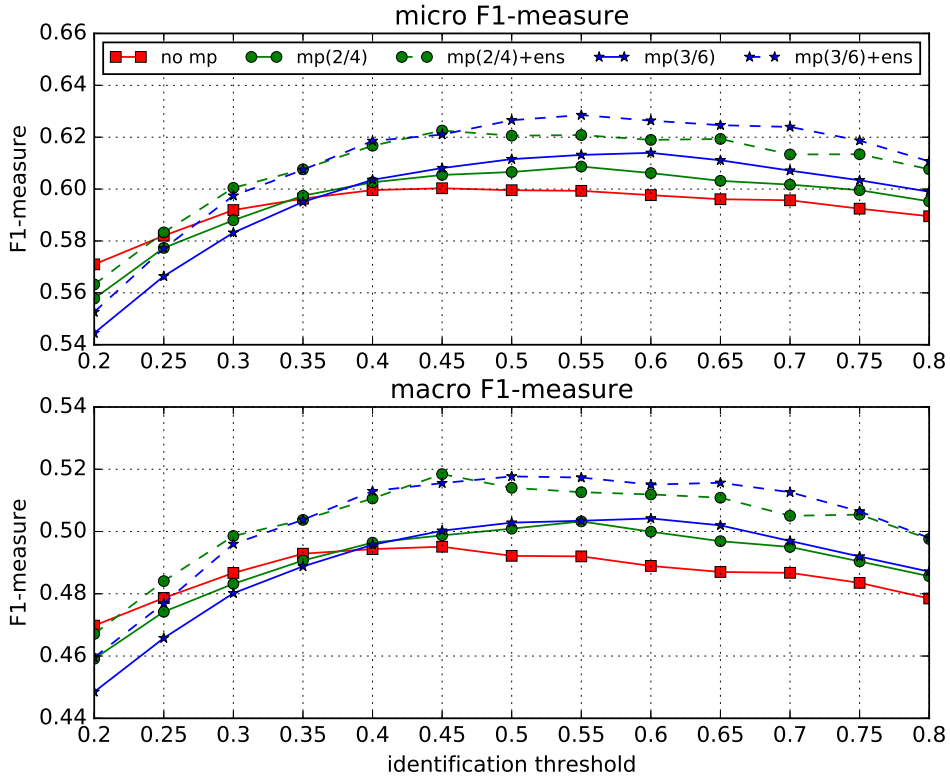

Fig. 5.4 Micro and macro $F1$ measure of without max-pooling, max-pooling with 4 and 6 frame, and mean model ensemble cases according to the identification threshold.

each class in the sliding window. We tried empirically chosen sliding window size of 4 and 6, which are 2.5 s and 3.5 s in time under 1.0 s analysis window, with hop size of half window size for the experiment. We experimented with various identification thresholds as in the analysis window size comparison experiment, because optimal threshold value might differ for after applying max-pooling.

As a result, it was possible to observe that max-pooling improves the performance, and using a pooling window of 6 frames found to be an optimal size as demonstrated in Figure 5.4. Optimal identification threshold was about 0.45 for the case without max-pooling, but it is increased to 0.55. It demonstrates

Table 5.4 Instrument Recognition Performance of the Proposed ConvNet with Various Activation Functions. Note that these results are from ensemble model.

Activation func.	Micro			Macro		
	P	R	$F1$	P	R	$F1$
tanh	0.575	0.593	0.584	0.465	0.527	0.479
ReLU	0.657	0.603	0.629	0.540	0.547	0.517
PReLU	0.630	0.608	0.618	0.496	0.533	0.505
LReLU (α=0.01)	0.651	0.611	0.631	0.524	0.550	0.524
LReLU (α=0.33)	0.658	0.567	0.609	0.545	0.509	0.508

that max-pooling helped the instruments with sporadic activation to survive from the output averaging process such that higher identification threshold can be used. The best micro F1-measure without max-pooling was 0.600 ($\theta = 0.45$) and it is improved to 0.613 ($\theta = 0.55$), and the best for macro measure without max-pooling was 0.495 ($\theta = 0.45$) and it is improved to 0.504 ($\theta = 0.60$).

In addition, we also experimented with model ensemble technique as it is reported that combinations of several different predictors can improve performance compared to the case of using a single model. This is because results generated using exactly the same network may slightly differ, and a model ensemble can generalize this problem [104]. We combined outputs probabilities from three individual models by taking an average, which is simple and one of the most widely used model ensemble technique. As demonstrated in Figure 5.4, it was possible to obtain further performance improvements via the use of a model ensemble, improved from 0.613 to 0.629 for micro, from 0.503 to 0.517 for macro F1-measure under $\theta = 0.55$, which found to be an optimal identification threshold when max-pooling and model ensemble is used.

5.5.3 Effect of Activation Function

In the case of using rectified units as an activation function, it was possible to observe a performance improvement compared to the tanh baseline as expected, as shown in Table 5.4. However, unlike the results presented in the ImageNet classification work from He et al. [102] and empirical evaluation work on ConvNet activation function from Xu et al. [103], PRelu and LReLU did not show visible improvement from normal ReLU, but just showed nearly matched or slightly decreased performance.

In order to a statistical significance of this result, we performed one-way analysis of variance (ANOVA) on the obtained F1-measures, under the standard 0.05 significance level. Because the analysis requires multiple attempts for each member of the group, we used the result of three individual models for each activation function, not the ensemble model.

As a result, p-value for the case of using all five activation function was 0.017 for micro and 0.029 for macro F1-measure. This result rejects the null hypothesis which means that the differences between some of the means are statistically significant. It is obvious that tanh function is significantly worse than Relu and its variations, we repeated the analysis again excluding the result of tanh function. As a result, we obtained a p-value of 0.067 for micro and 0.440 for macro F1-measure. This result indicates that using modified version of ReLU did not show the statistically significant difference in our experiment, unlike for computer vision tasks. Following the result of analysis, we decided to use normal ReLU as an activation function for our final model.

5.5.4 Comparison to Existing Algorithms

We could find the optimal setting for our ConvNet from the various experiments demonstrated above. For analysis window and identification threshold θ, 1.0 s and 0.55 found to be an optimal value, respectively. We decided to use normal ReLU as an activation function because we could not find statistically significant improvement from variants of ReLU, and class-wise max-pooling was applied on sigmoid outputs prior to output aggregation. Finally, we averaged output probabilities of three individual models for a model ensemble. As mentioned above, these experiments were conducted using development set. To compare the performance with other existing algorithms, we tested our network on the separate test set which is not used for training and development. For the result, our network achieved 0.619 for the micro $F1$ measure and 0.513 for the macro $F1$ measure on the test set. It is very close to the results obtained from the development set which are 0.629 and 0.517 respectively, and this result shows that our proposed method is less likely over-fitted to the data, because development set and test set contains audio clips from different songs which covers a wide variability in recording and production styles, also from various decades.

The existing algorithm from Fuhrmann and Herrera [105] used typical handmade timbral audio features with their frame-wise mean and variance statistics to train SVMs, and Bosch et al. [77] improved this algorithm with source separation called FASST (Flexible Audio Source Separation Framework) [106] in a preprocessing step.

As shown in Figure 5.5, our proposed system marginally outperformed existing algorithms. In terms of precision, Fuhrmann and Herrera's algorithm showed the best performance for both the micro and the macro measure. However, its

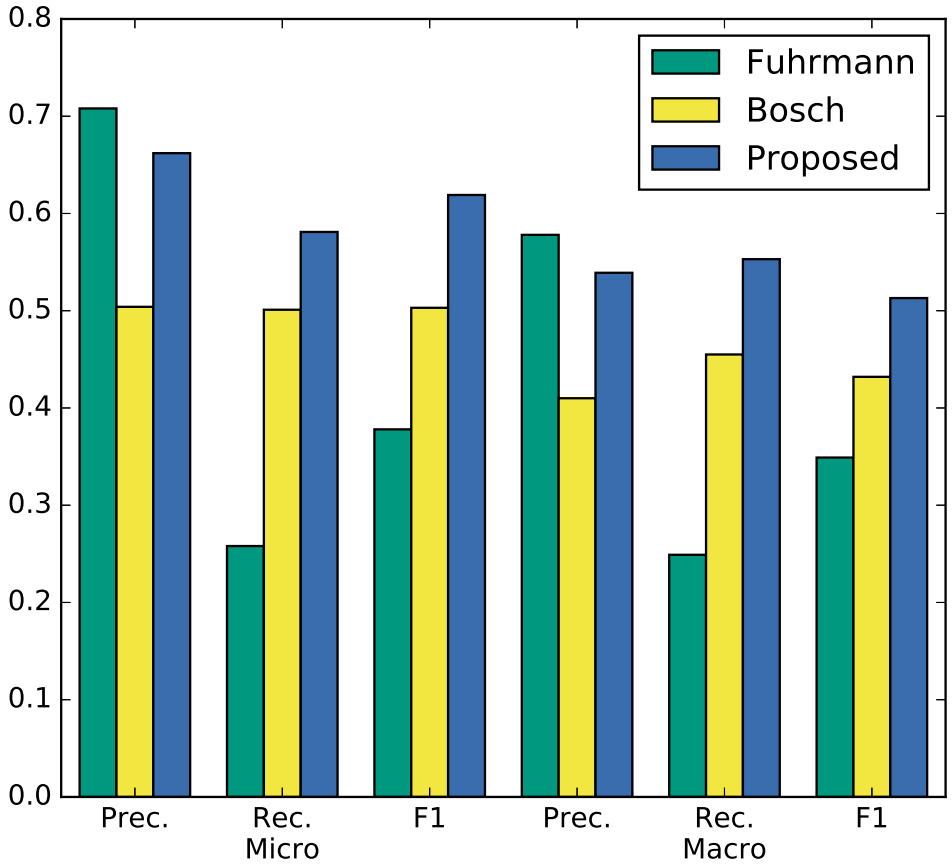

Fig. 5.5 Performance comparison of the predominant instrument recognition algorithm from Fuhrmann and Herrra [105], Bosch et al. [77], and our proposed ConvNet. Note that the result of proposed algorithm used half of the original test set, because another half was used for the development.

recall was very low, around 0.25, which resulted in a low $F1$ measure. From this result, it can be observed that the learned feature from the input data that is classified through ConvNet works better than the conventional hand-crafted features with SVMs.

5.5.5 Analysis of Instrument-Wise Identification Performance

The results demonstrated above were focused on the overall identification performance. In this section, we analyze and discuss the result instrument-wise (i.e., class-wise) to observe the system performance in detail. As shown in Figure 5.6, identification performance varies to a great extent, depending on the instruments. It can be observed that the system recognizes the voice in the music very well, showing an $F1$ measure of 0.821. On the other hand, cello and clarinet showed relatively poor performance compared to other instruments, showing an $F1$ measure of around 0.10.

One of the reasons for poor performance of cello and clarinet is highly likely the insufficient number of testing audio samples. The dataset only has 111 and 62 test audio samples for cello and clarinet, respectively, which are the first and second least number of test audio, while it has 1044 audio samples for the human voice. Evaluating the system on a small number of test data would make the result less reliable and less stable than other identification results. In addition, our training examples for the voice was more than twice bigger than that of cello, and this data imbalance is also likely one of the main reasons for the performance degrade. It is reported that data imbalance causes the distortion of decision boundaries and the decision boundary of majority class tends to invade that of minority class [107].

Apart from the issue related to the number of audio examples, high identification performance of the voice class is highly likely owing to its spectral characteristic that is distinct from other musical instruments. The other instruments used in the experiment usually produce relatively clear harmonic patterns; however, the human voice produces highly unique spectral characteristics that contain much more inharmonic spectrum with a natural vibrato.

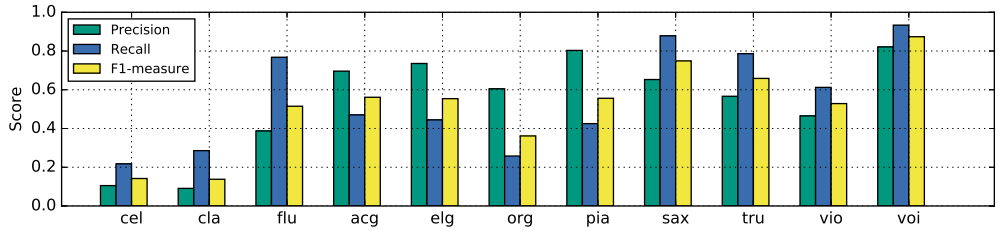

Fig. 5.6 Precision, recall, and F1-measure of each instrument with the optimal parameters.

On the other hand, identification performance for flute was moderate, although it also has relatively small testing examples. It seems it is also due to highly sine-wave-like spectral characteristics of flute, which clearly differs from other instruments.

Although we use F1-measure as the main performance measure, it is intriguing to observe the precision and recall for each instrument. Instruments such as acoustic guitar, electric guitar, organ, and piano showed higher precision compare to recall, while other instruments showed relatively higher recall compare to precision. We found that these can be categorized into two groups, which are instruments with the hard onset and soft onset. A hard onset is accompanied by a sudden change, whereas a soft onset shows a relatively gradual change in energy [108]. It was possible to observe that instruments with higher precision are usually played with hard onset while with higher recall are played with soft onset. Because we used the same threshold for all instruments, the result can be understood to mean that our threshold was relatively strict for hard onset instruments but loose for the instruments with soft onset. Hence, this result shows that there is potential performance improvement by applying a different identification threshold according to the type of onset.

This result can be interpreted as spectral characteristics of hard onset instruments at the exact onset moment was clearer than the one of soft onset,

hence the network could identify them easier. On the other hand, soft onset instruments do not have strong energy on the onset moment, and it means that the system identifies instruments mostly based on the sustain part of the sound. It implies that onset is an important clue for judging predominant instrument in the audio clips.

5.5.6 Analysis on Single Predominant Instrument Identification

Throughout the paper, we focused on identifying multiple predominant instruments exist in the music. However, we believe that it is also valuable to observe the performance of our proposed network applied onto the music with single predominant instrument to investigate a 'pure' performance of the network. To this end, we divided our training data into halves and used one for training, another one for testing the network. We divided the data into two chunks before slicing them into 1.0 s, thus no music from training data were included in testing data. Because there were no multiple instruments in this experiment, we simply took the instrument with the highest probability as a predominant instrument.

As a result, we obtained an average accuracy of 0.633, and the confusion matrix for 11 instruments was demonstrated in Figure 5.7. Identification performances were relatively evenly spread over all classes compare to the multiple predominant instrument case. For instance, performances of cello/clarinet and voice were moderate while they showed a big performance gap for multiple instrument case. On the other hand, saxophone showed the worst accuracy while it showed the second best performance for multiple instrument case. This result shows that the performance for the single instrument case does not always match the performance for multiple instruments case, because sound of several

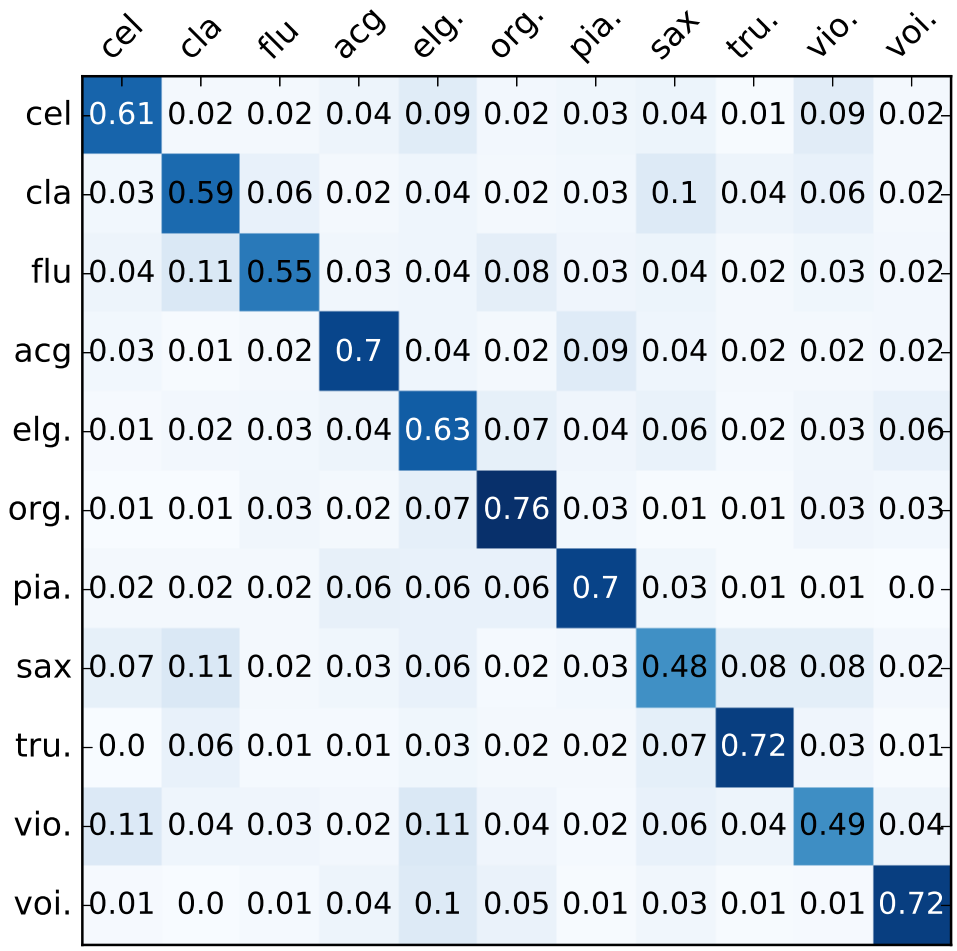

Fig. 5.7 Confusion matrix for single predominant instrument identification. X axis indicates predicted label and Y axis indicates true label.

instruments are overlapped. Also, it seems the aggregation process makes huge difference on the final performance. Note that because we used only half of the training set for this analysis, performance of the actual model used for the multiple predominant instrument would be better than the result demonstrated in this section to some extent.

5.5.7 Qualitative Analysis with Visualization Methods

To understand the internal mechanism of the proposed model, we conducted a visual analysis with various visualization methods. First, we tried clustering for each layer's intermediate hidden states from a given input data sample to verify how the encoding behavior of each layer contributes to the clustering of input samples. We selected the t-distributed stochastic neighbor embedding (t-SNE) [109] algorithm, which is a technique for dimensionality reduction of high-dimensional data. Second, we exploited the deconvolution [93, 110] method to identify the functionality of each unit in the proposed ConvNet model by visual analysis. Our system basically repeats two convolutional layers followed by one pooling layer, and we grouped these three components and call it "convolutional block" throughout this section for simplicity.

The t-SNE algorithm is based on the stochastic neighbor embedding (SNE) algorithm, which converts the similarities between given data points to joint probability and then embeds high-dimensional data points to lower-dimensional space by minimizing the Kuller-Leibler divergence between the joint probability of low-dimensional embedding and the high-dimensional data points. This method is highly effective especially in a dataset where its dimension is very high [109]. This advantage of the algorithm accorded well with our condition, where the target observations were necessarily in a high dimension since we reshaped each layer's filter activations to a single vector respectively.

With the visualization exploiting t-SNE, we could observe how each layer contributed to the classification of the dataset. Reflecting a gradually changing inter-distance of data points at each stage of the proposed model, four intermediate activations were extracted at the end of each convolutional block and one from the hidden fully connected layer, and another one from the final out-

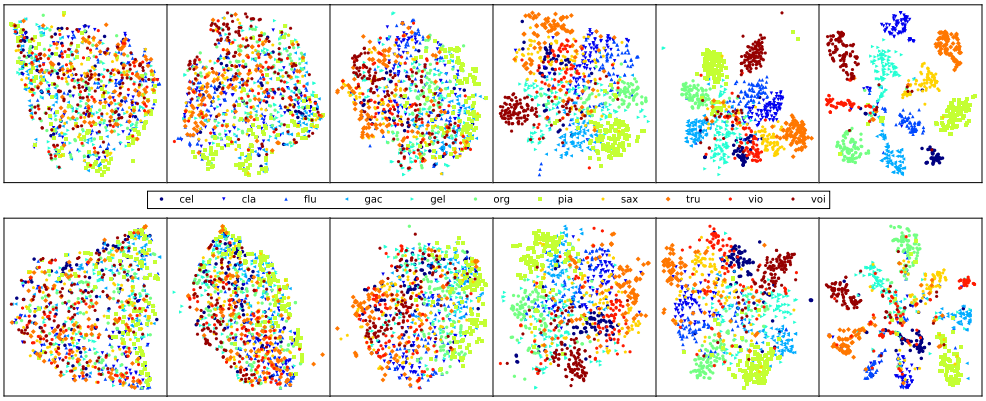

Fig. 5.8 Visualization of the t-SNE clustering result. It represents the clustering result for each intermediate state of the proposed model. From left to right, the first four plots are the clustering result of the activations at the end of each convolutional block, and the last two plots are the clustering result of the activations of the hidden dense layer and the final sigmoid output, respectably. The upper plots are drawn from the sample used in the training, and the lower plots are from the validation data samples.

put layer. For the compression of dimensionality and computational efficiency, we pooled the maximum values for activation matrices of each unit. By this process, the dimensionality of each layer's output could be diminished to each layer's unit size. We visualized on both randomly selected training and validation data samples from the entire dataset to verify both how the model exactly works and how it generalizes its classification capability. In Figure 5.8, it is clearly shown that data samples under the same class of instrument are well grouped and each group is separated farther, with the level of encoding being higher, particularly on the training set. While the clustering was not clearer than the former case, the tendency of clustering on the validation set was also found to be similar to the training set condition.

Another visualization method, deconvolution, has recently been introduced as a useful analysis tool to qualitatively evaluate each node of a ConvNet. The

main principle of this method is to inverse every stage of operations reaching to the target unit, to generate a visually inspectable image that has been, as a consequence, filtered by the trained sub-functionality of the target unit [93]. With this method, it is possible to reveal intuitively how each internal sub-function works within the entire deep convolutional network, which tends to be thought of as a "black box".

By this process, the functionality of a sub-part of the proposed model is explored. We generated deconvoluted images like those in Figure 5.9 from the arbitrary input mel-spectrogram, for each unit in the entire model. From the visual analysis of the resulting images, we could see several aspects of the sub-functionalities of the proposed model: (1) Most units in the first layer tend to extract vertical, horizontal, and diagonal edges from the input spectrogram, just like the lower layers of ConvNets do in the usual image object recognition task. (2) From the second layer through the fourth layer, each deconvoluted image indicates that each unit of the mid-layers has a functionality that searches for particular combinations of the edges extracted from the first layer. (3) It was found that it is difficult to strongly declare each sub-part of the proposed model that detects a specific musical articulation or expression. However, in an inductive manner, we could see that some units indicate that they can be understood as a sub-function of such musical expression detector.

We conducted a visual analysis of the deconvoluted image of two independent music signals, which have the same kind of sound sources, but differently labeled.[1] For both cases, the most activated units of the first layer strongly suggested that their primary functionality is to detect a harmonic component on the input mel-spectrogram by finding horizontal edges in it, as shown in

[1]Both signals were composed of a "voice" and an "acoustic guitar" instrument, but the predominant instrument of signal (A) was labeled as the "voice," while (B) was labeled as the "acoustic guitar".

the top figures in Figure 5.9. However, from the second layer to higher layers, the highly activated units' behavior appeared to be quite different for each respective input signal. For instance, the most activated unit of signal (A)'s second layer showed a functionality similar to onset detection, by detecting a combination of vertical and horizontal edges. Compared to this unit, the most activated units of the third layer showed a different functionality that seems to activate unstable components such as the vibrato articulation or the "slur" of the singing voice part, by detecting a particular combination of diagonal and horizontal edges. On the other hand, the model's behavior in signal (B) was very different. As is clearly shown in the second and the third layers' output in Figure 5.9, the highly activated sub-functions were trying to detect a dense field of stable, horizontal edges which are often found in harmonic instruments like guitar. Each field detected from those units corresponded to the region where the strumming acoustic guitar sound is.

5.6 Conclusions

In this paper, we described how to apply ConvNet to identify predominant instrument in the real-world music. We trained the network using fixed-length single-labeled data, and identify an arbitrary number of the predominant instrument in a music clip with a variable length. Our results showed that very deep ConvNet is capable of achieving good performance by learning the appropriate feature automatically from the input data. Our proposed ConvNet architecture outperformed previous state-of-the-art approaches in a predominant instrument identification task on the IRMAS dataset. Mel-spectrogram was used as an input to the ConvNet, and we did not use any source separation in the preprocessing unlike in existing works.

We used sliding window to perform short-time analysis, and obtained sigmoid outputs were aggregated by taking class-wise average so that the system can handle music excerpts with various lengths. Then, we normalized and threshold them to find predominant instruments. We conducted an extensive experiment with various analysis window sizes and identification thresholds. For the result, 1.0 s was found to be the optimal window size and a threshold value of 0.55 showed the best performance. Also, we conducted an experiment with applying max-pooling on the output aggregation process. By applying max-pooling, it was possible to help the instrument with sporadic activation not to be suppressed by the audio clip-wise averaging process, hence the performance is improved further. We also conducted several experiments with various activation functions for ConvNet. Results confirmed that ReLU worked reasonably well, which is a de facto standard in recent ConvNet studies. However, we could not find statistically significant improvement by replacing it to LReLU or PReLU. Using the optimal parameters we found, we achieved 0.619 for micro and 0.513 for macro F1-measure which outperforms existing algorithm on predominant instrument identification task.

In addition, we demonstrated several extra experiments for analyzing obtained results. Our analysis on the instrument-wise identification result found that identification performance for each instrument varies to a great extent. Cello and clarinet showed very low identification result while voice is well recognized. Also, we found that the network usually showed high precision with low recall for hard onset instruments, high recall with low precision for soft onset instruments. Visualization of the intermediate outputs using t-SNE showed that the feature representation became clearer each time the input data were passed through the convolutional blocks. Moreover, visualization using deconvolution showed that the lower layer tended to capture the horizontal and vertical

edges, and that the higher layer tended to seek the combination of these edges to describe the spectral characteristics of the instruments.

Our study shows that many recent advances in a neural network in the image processing area are transferable to the audio processing domain. However, audio signal processing, especially music signal processing, has many different aspects compared to the image processing area where ConvNets are most extensively used. For example, spectral characteristics are usually overlapped in both time and frequency unlike the objects in an image, which makes the detection difficult. Moreover, music signals are much more repetitive and continuous compared to natural images and are present in various lengths. Hence, we believe that applying musical domain knowledge on the system is highly important. For example, we found that hard onset instruments usually showed high precision compare to recall, and the soft onset instruments showed high recall compare to precision. We are planning to apply a different threshold depending on onset type of the instrument, and such domain knowledge is highly helpful to improve the performance further.

Furthermore, it is recently reported that deep neural networks are vulnerable to adversarial examples [111]. This means that the network can misclassify examples that are only slightly modified, even the differences are indistinguishable to the human eye. We believe that it would be beneficial to train our model on a mixture of adversarial and clean examples in the future, for the better regularization.

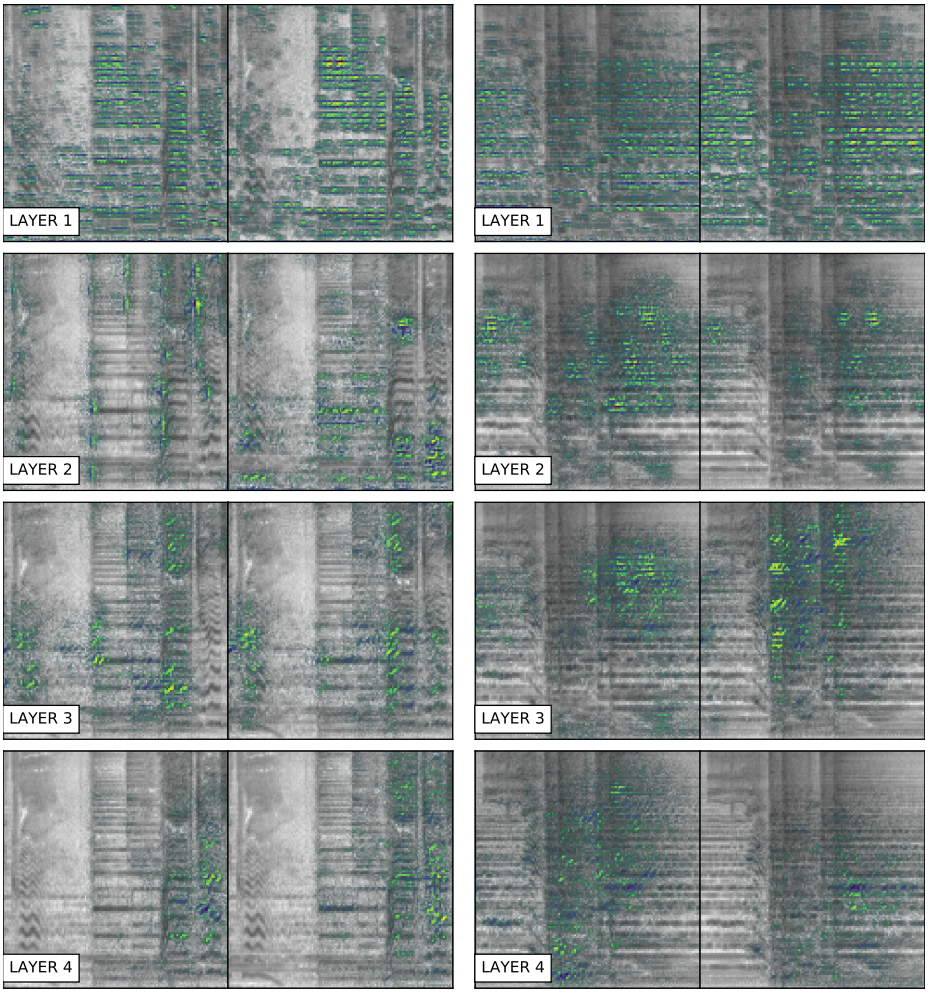

Fig. 5.9 Mel-spectrogram of two input signals and their respective deconvoluted results. (A) and (B) were calculated from two independent music signals randomly cropped from the original music and consist mainly of the voice and the acoustic guitar, where (A) is labeled as the voice and (B) as the acoustic guitar. Each row of images represents a deconvoluted signal overlaid on the original signal. We extracted these results from four intermediate stages of the proposed model. Deconvolution outputs were extracted from the end of each convolutional block. Two columns represent two highest activated units of each point. The green and navy color indicate the positive and negative part of the result, respectably. The range of activation result is normalized for the purpose of clear visualization.

Chapter 6

Conclusions and Future Work

The aim of the thesis was to improve the state-of-the-art performance of musical instrument related MIR tasks such as tone detection and instrument identification. Once these information are extracted from the audio, it can be used for various music-related applications. In the music education field, tone detection allows the system to return feedback such as fingering mistakes, and instrument identification allows the system to provide appropriate style of musical score according to the instrument without manual input. In addition, if instrument information can be extracted from the polyphonic real-world music, people can search and browse the music by instrument which is not possible at the moment.

Conventional approaches to these tasks use hand-crafted features from the audio and decision is typically made by classifiers such as SVMs and RF. However, learned features recently have outperformed traditional hand-crafted features for various MIR tasks. We used sparse feature learning and convolutional neural network for feature learning, which aims to learn an appropriate feature

for the target task using sparsity and local correlation, respectively. In this paper, We investigated musical instrument related MIR tasks in three different levels, start from the relatively simpler case and gradually extend the scope to the more complicated situation.

Firstly, we began with the flute fingering detection task. Overblown flute tones are highly similar to normally blown tone, but their timbres are slightly different. By recognizing various same-pitched overblown flute tones and normally blown sound, we could estimate the fingerings of the flute performer from the sound. We utilized sparse feature learning method and used mel-spectrogram of the audio as input data. It outperformed conventional acoustic features such as MFCCs, also residual noise spectrum with PCA/LDA which is previous state-of-the-art in this task. In addition, we could observe the spectral difference between overblown tones and normally blown tones by visualizing feature activation and learned basis, which makes an actual timbral difference.

Secondly, we investigated instrument identification in the monophonic situation. We used studio recorded solo tones of 24 instruments which cover whole pitch range of each instrument at half-tone intervals with various playing styles. We learned feature representation using sparse filtering as in the flute fingering detection task, but we found that using linear-scale spectrogram as an input feature representation of the audio significantly outperforms the case of using mel-scale spectrogram. The main contribution of this work is proposing frame sampling and pooling method to improve accuracy further. We proposed two different sampling method which is proportional and fixed sampling to examine whether there is a certain minimum amount of samples to learn a good feature representation or it should be determined by the length of the input audio. From the experiment, we found that the amount of frame sampling should be relative to the note length. Regarding pooling method, we found that SD-pooling

marginally outperforms widely used maximum and average pooling.

Lastly, we extend our scope to predominant instrument classification in a polyphonic situation. Also, we used real-world music, not a studio recorded solo tones. We utilized ConvNet for classifying 11 instruments, and the network was trained using fixed length single-labeled data while test data is variable length multi-labeled. Unlike previous two experiments using sparse filtering, we learned features in a supervised manner using ConvNets. We conducted an extensive experiment with various settings and found that using 1 s for analysis window is the optimal for the task. In addition, we investigated the aggregation method to analyze audio with variable length. We found that taking average of output probabilities over time axis followed by normalization works well, and it was possible to improve the performance further by applying max-pooling on the output probabilities before averaging to prevent sporadically appearing instruments suppressed by the averaging process. As a result, proposed approach achieved state-of-the-art performance by outperforming previous works using hand-made audio features and source separation techniques. We also visualized how each convolution layer works using deconvolution method as well as how the feature representation gets better as the data pass through convolutional layers using the t-SNE algorithm.

In this thesis, we have investigated the optimal method to apply feature learning technique on the sound of musical instrument. As a consequence, we found that the keys of successful application of audio feature learning can be summarized to several critical factors such as appropriate input feature representation, learning algorithm, network architecture, and activation pooling. In the future, we would like to improve the performance using the concept of deep residual learning [47] and batch normalization [112] technique which recently have shown the superior result on various computer vision tasks. In addition,

we believe that there is a space for applying recurrent neural networks (RNNs) technique in output aggregation process to make use of temporal information.

Our main goal was to automatically identify the sound of musical instrument. However, we found that our ConvNet approach used for instrument identification on polyphonic music can be applied to other audio processing tasks with just small modifications. We have participated Detection and Classification of Acoustic Scenes and Events (DCASE 2016) challenge organized by IEEE Audio and Acoustic Signal Processing (AASP) Technical Committee and details of our approach can be found in [113]. We used similar ConvNet framework with that of instrument identification, but added multiple-width frequency-delta (MWFD) data augmentation which emphasizes spectral characteristics of acoustic events in a various delta resolution. We are planning to keep investigate on a method that can be generally applied to other audio processing tasks.

Although the optimal network architecture and data processing method would be highly depending on target tasks, we believe that the result obtained in this thesis can be a good starting point for someone who aims to apply feature learning techniques for audio processing, also beneficial for improving existing audio feature learning framework. Below is summary of what we found through the researches and experiments, based on what we consider important factors of audio feature learning which are illustrated in Fig. 1.6.

Audio preprocessing

Mono or stereo We used mono input for all experiments which is easier to handle. However, we think that using stereo have great potential especially when the task is related to environmental sounds such as acoustic scene and event detection.

Sampling rate 44,100 Hz is highly common audio sampling rate, but we used downsampled it to 22,050 Hz for the experiments in this thesis. For MIR tasks, we think 22,050 Hz is usually a sufficient sampling rate to preserve important information while reducing the data size. For speech, lower sampling rate such as 12,000 or 8,000 Hz would be better because high frequency range which does not include the information about the speech itself may lead the system to be overfitted to the recording environment of training data. For acoustic scene and event detection, we obtained the better result with full 44,100 Hz which is reasonable because certain spectral characteristics of environmental sound might exist in a very high frequency range.

Source separation We did not use source separation technique for the proposed system. However, we think that it would be interesting to apply source separation for removing percussive sound or emphasizing harmonic components.

Recording environment It is highly important to aware that recording environment is one of the most crucial factors that cause overfitting for audio feature learning. Recording environment here means not only the location, but also the recording device. It is better to use audio from the various environment and make sure the validation and training set does not contain the audio from the same recording environment.

Window size Images are usually nearly square sized, but time dimension of the audio signal is much longer than the feature dimension depending on the length of the audio clip. Hence, it is common to use windows to cut audio clips into multiple chunks. The optimal window size is different depending on tasks, but we found that 1 s is a good starting point for the experiment, especially for ConvNet.

Input representation

Time domain or time-frequency domain We used Fourier transform for convert time-domain signal to the frequency domain for all experiments. However, increasing number of researchers attempt to learn a feature from time-domain signal directly. This topic is discussed at the end of this section.

Frequency scale We used linear-scale spectrogram or mel-scale spectrogram in the experiments. We found that 128 bin mel-scale spectrogram (under 22,050 Hz sampling rate) is good starting point for various tasks. However, our monophonic instrument classification experiment has shown that if spectral characteristics of some of the target labels are too similar, it is better not to use mel-scale for dimension reduction because it might not provide sufficient spectral resolution.

Data augmentation Using data augmentation can be highly helpful to increase the number of training examples, to make the system robust to noises, also to find a hidden pattern exist in the data. We used delta feature in spectral axis for MWFD data augmentation explained above in this section, and found that it visibly improved the performance. We think that delta features in frequency/temporal axis are highly valuable, also data augmentation using multiple frequency/temporal resolution would be helpful regardless of tasks.

Network architecture and learning

Shallow or deep architecture Deeper network architecture can extract higher level information, but it does not mean it is always better. In many cases, shallower network works better because very deep network architecture

easily overfits to training data unless the amount of training data is very huge. Hence, it is beneficial to try removing or adding some layers.

Convolutional or recurrent ConvNet and RNN are highly popular deep learning algorithms. ConvNet is usually working well for timbral classification tasks, or when audio clip includes some acoustic events that can be a 'clue' for the classification. RNN is obviously working well for when the sequential information is important. However, we believe that RNN can be used along with the ConvNet to improve the performance further because audio always contains sequential information.

Pooling and subsampling Pooling is an intuitive and effective method to add extra robustness on the system while saving the computation power. For MIR applications, it is important to consider temporal resolution when applying pooling. Although max-pooling is the most popular choice in computer vision field, but we found that SD-pooling extracts intriguing information from the audio, especially for wind instruments.

Regularization and hyperparameters Most of the regularization techniques for deep learning are not domain specific. Dropout is always good choice to put after convolution or recurrent layers, but dropout rate should be depending on the dataset size. If a huge amount of data is available for training, removing dropout would be fine. Recently, batch normalization became one of the most popular choices for regularization. We think that for ConvNets, it is always better to include batch normalization, although it is not needed for RNN.

For deep learning, there are a number of hyperparameters and many of them are empirically selected. Rather than trying every possible value, we would

suggest using the similar setting with popular networks and start fine tuning from there.

Postprocessing

Output aggregation As mentioned above, audio data are usually divided into multiple chunks to be processed, and length of the audio might differ clip to clip. In this case, aggregation of window-scale results is needed to obtain an audio clip-scale result. We found that simply using average probability works reasonably well for this purpose. This kind of postprocessing is usually not required for computer vision, because fixed sized images are used or it is possible to crop or resize images to have identical size. We believe that using RNN instead of average probability for the output aggregation process would be another sensible approach if the number of training example is large enough.

Model ensemble or network fusion The model ensemble is a process that uses the results of multiple networks to gain an extra performance improvement. Although technique itself is not task or domain specific, the effect of ensemble might differ depending on the task. Taking an average of several output probabilities is called mean ensemble, and it is definitely the most popular choice and guarantees at least tiny bit of performance improvement. We found that using other choices like median or other statistics sometimes works better, but it is highly depending on the task. Due to randomness in training, models for ensemble can be simply just same network trained again. To obtain full potential of the model ensemble, it is beneficial to vary the network depth or dropout rate to introduce more 'disagreement' among output probabilities. The more aggressive way of using multiple networks at once would be network fusion. Unlike model ensemble, this term usually indicates a network

that contains multiple network inside of on huge network connected and trained simultaneously. This method is usually computationally heavy, does not guarantee performance gain, also difficult to get it trained, although it might provide a much better result when the architecture is designed well. We think that models trained on different preprocessing methods with different algorithms would have a great synergy when they are trained together as a fusion network.

As explained above, simply putting data into well-known machine learning algorithms is not sufficient for successful audio feature learning. In addition, many deep learning techniques are transferable from computer vision field, but there are many 'domain-specific' parts that should be taken into account when designing a system. We believe that making a baseline method for such domain-specific parts is highly helpful for other researchers, and we wish that our research made a meaningful contribution on this matter.

Regarding input data representation, we used time-frequency representation such as mel-spectrogram or linear-scale spectrogram depending on the target task. However, DFT step for convert time domain audio into time-frequency representation is one of the hurdles for real-world application because commercial DFT libraries are quite expensive. There are a number of recent attempts at using raw audio data for the network input, called end-to-end learning, but the performance is still slightly below that of spectral input [14, 15]. If time-domain audio can be used as an input representation for feature learning, this would make real-world implementation much easier and cheaper. Hence, we are going to keep investigate on a network architecture that can learn features directly from a time-domain audio signal while outperforms time-frequency representation.

References

[1] I. Godt, "Music: A practical definition," *The Musical Times*, vol. 146, no. 1890, pp. 83–88, 2005.

[2] T. Kitahara, M. Goto, and H. G. Okuno, "Musical instrument identification based on f0-dependent multivariate normal distribution," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, vol. 5. IEEE, 2003, pp. V–421.

[3] A. Eronen and A. Klapuri, "Musical instrument recognition using cepstral coefficients and temporal features," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, vol. 2. IEEE, 2000, pp. II753–II756.

[4] A. Diment, P. Rajan, T. Heittola, and T. Virtanen, "Modified group delay feature for musical instrument recognition," in *10th International Symposium on Computer Music Multidisciplinary Research (CMMR). Marseille, France*, 2013.

[5] D. Bhalke, C. R. Rao, and D. Bormane, "Musical instrument classification using higher order spectra," in *Signal Processing and Integrated Networks (SPIN), 2014 International Conference on.* IEEE, 2014, pp. 40–45.

[6] M. J. Newton and L. S. Smith, "A neurally inspired musical instrument classification system based upon the sound onset," *The Journal of the Acoustical Society of America*, vol. 131, no. 6, pp. 4785–4798, 2012.

[7] A. Krishna and T. V. Sreenivas, "Music instrument recognition: from isolated notes to solo phrases," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, vol. 4. IEEE, 2004, pp. iv–265.

[8] S. Essid, G. Richard, and B. David, "Musical instrument recognition on solo performances," in *Signal Processing Conference, 2004 12th European.* IEEE, 2004, pp. 1289–1292.

[9] T. Heittola, A. Klapuri, and T. Virtanen, "Musical instrument recognition in polyphonic audio using source-filter model for sound separation." in *ISMIR*, 2009, pp. 327–332.

[10] T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno, "Instrument identification in polyphonic music: Feature weighting to minimize influence of sound overlaps," *EURASIP Journal on Applied Signal Processing*, vol. 2007, no. 1, pp. 155–155, 2007.

[11] Z. Duan, B. Pardo, and L. Daudet, "A novel cepstral representation for timbre modeling of sound sources in polyphonic mixtures," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on.* IEEE, 2014, pp. 7495–7499.

[12] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "Rwc music database: Music genre database and musical instrument sound database." in *ISMIR*, vol. 3, 2003, pp. 229–230.

[13] A. De Cheveigné and H. Kawahara, "Yin, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[14] Y. Hoshen, R. J. Weiss, and K. W. Wilson, "Speech acoustic modeling from raw multichannel waveforms," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4624–4628.

[15] D. Palaz, R. Collobert, and M. M. Doss, "Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks," *arXiv preprint arXiv:1304.1018*, 2013.

[16] R. N. Bracewell, "Fourier transform and its applications," 1999.

[17] B. Mulgrew, P. M. Grant, and J. Thompson, *Digital signal processing: concepts and applications*. Macmillan Press, 1999.

[18] R. Kronland-Martinet, J. Morlet, and A. Grossmann, "Analysis of sound patterns through wavelet transforms," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 1, no. 02, pp. 273–302, 1987.

[19] G. Li and A. A. Khokhar, "Content-based indexing and retrieval of audio data using wavelets," in *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, vol. 2. IEEE, 2000, pp. 885–888.

[20] G. Tzanetakis, G. Essl, and P. Cook, "Audio analysis using the discrete wavelet transform," in *Proc. Conf. in Acoustics and Music Theory Applications*, 2001.

[21] R. Kronland-Martinet, "The wavelet transform for analysis, synthesis, and processing of speech and music sounds," *Computer Music Journal*, vol. 12, no. 4, pp. 11–20, 1988.

[22] B. Logan *et al.*, "Mel frequency cepstral coefficients for music modeling." in *ISMIR*, 2000.

[23] A. M. Noll, "Cepstrum pitch determination," *The journal of the acoustical society of America*, vol. 41, no. 2, pp. 293–309, 1967.

[24] B. P. Bogert, M. J. Healy, and J. W. Tukey, "The quefrency alanysis of time series for echoes: Cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking," in *Proceedings of the symposium on time series analysis*, vol. 15. chapter, 1963, pp. 209–243.

[25] J. Hillenbrand, L. A. Getty, M. J. Clark, and K. Wheeler, "Acoustic characteristics of american english vowels," *The Journal of the Acoustical society of America*, vol. 97, no. 5, pp. 3099–3111, 1995.

[26] L.-F. Yu, L. Su, and Y.-H. Yang, "Sparse cepstral codes and power scale for instrument identification," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on.* IEEE, 2014, pp. 7460–7464.

[27] Y. Pan and A. Waibel, "The effects of room acoustics on mfcc speech parameter." in *INTERSPEECH*, 2000, pp. 129–132.

[28] A. M. Noll, "Pitch determination of human speech by the harmonic product spectrum, the harmonic sum spectrum, and a maximum likelihood estimate," in *Proceedings of the symposium on computer processing communications*, vol. 779, 1969.

[29] E. Schubert and J. Wolfe, "Does timbral brightness scale with frequency and spectral centroid?" *Acta acustica united with acustica*, vol. 92, no. 5, pp. 820–825, 2006.

[30] A. Klapuri and M. Davy, *Signal processing methods for music transcription*. Springer Science & Business Media, 2007.

[31] R. Rojas, *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[33] J. Ngiam, Z. Chen, S. A. Bhaskar, P. W. Koh, and A. Y. Ng, "Sparse filtering," in *Advances in neural information processing systems*, 2011, pp. 1125–1133.

[34] M. Henaff, K. Jarrett, K. Kavukcuoglu, and Y. LeCun, "Unsupervised learning of sparse features for scalable audio classification." in *ISMIR*, vol. 11, no. 445, 2011, p. 2011.

[35] J. Schluter and C. Osendorfer, "Music similarity estimation with the mean-covariance restricted boltzmann machine," in *Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on*, vol. 2. IEEE, 2011, pp. 118–123.

[36] J. Nam, J. Ngiam, H. Lee, and M. Slaney, "A classification-based polyphonic piano transcription approach using learned feature representations." in *ISMIR*, 2011, pp. 175–180.

[37] J. Nam, J. Herrera, M. Slaney, and J. O. Smith, "Learning sparse feature representations for music annotation and retrieval." in *ISMIR*, 2012, pp. 565–570.

[38] J. Schluter and S. Bock, "Improved musical onset detection with convolutional neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on.* IEEE, 2014, pp. 6979–6983.

[39] E. J. Humphrey and J. P. Bello, "Rethinking automatic chord recognition with convolutional neural networks," in *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, vol. 2. IEEE, 2012, pp. 357–362.

[40] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Audio chord recognition with recurrent neural networks." in *ISMIR*, 2013, pp. 335–340.

[41] K. Ullrich, J. Schlüter, and T. Grill, "Boundary detection in music structure analysis using convolutional neural networks." in *ISMIR*, 2014, pp. 417–422.

[42] T. Grill and J. Schlüter, "Music boundary detection using neural networks on combined features and two-level annotations," in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR 2015), Malaga, Spain*, 2015.

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

[44] L. Deng and D. Yu, "Deep learning: Methods and applications," *Foundations and Trends in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[45] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[46] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[48] 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, 2008.

[49] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D. J. Wu, and A. Y. Ng, "Text detection and character recognition in scene images with unsupervised feature learning," in *2011 International Conference on Document Analysis and Recognition*. IEEE, 2011, pp. 440–445.

[50] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[51] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceed-*

ings of the 25th international conference on Machine learning. ACM, 2008, pp. 1096–1103.

[52] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[53] M. Schmidt, "minfunc: unconstrained differentiable multivariate optimization in matlab," *URL http://www. di. ens. fr/mschmidt/Software/minFunc. html*, 2012.

[54] Y. Han and K. Lee, "Hierarchical approach to detect common mistakes of beginner flute players." in *ISMIR*, 2014, pp. 77–82.

[55] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[56] Y. LeCun, L. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard *et al.*, "Learning algorithms for classification: A comparison on handwritten digit recognition," *Neural networks: the statistical mechanics perspective*, vol. 261, p. 276, 1995.

[57] A. Calderón, S. Roa, and J. Victorino, "Handwritten digit recognition using convolutional neural networks and gabor filters," *Proc. Int. Congr. Comput. Intell*, 2003.

[58] X.-X. Niu and C. Y. Suen, "A novel hybrid cnn–svm classifier for recognizing handwritten digits," *Pattern Recognition*, vol. 45, no. 4, pp. 1318–1325, 2012.

[59] J. Ngiam, Z. Chen, D. Chia, P. W. Koh, Q. V. Le, and A. Y. Ng, "Tiled convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2010, pp. 1279–1287.

[60] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[61] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[62] P. Hamel, S. Lemieux, Y. Bengio, and D. Eck, "Temporal pooling and multiscale learning for automatic annotation and ranking of music audio." in *ISMIR*, 2011, pp. 729–734.

[63] J. W. Coltman, "Sounding mechanism of the flute and organ pipe," *The Journal of the Acoustical Society of America*, vol. 44, no. 4, pp. 983–992, 1968.

[64] J. Wolfe, J. Smith, J. Tann, and N. H. Fletcher, "Acoustic impedance spectra of classical and modern flutes," *Journal of sound and vibration*, vol. 243, no. 1, pp. 127–144, 2001.

[65] J. Smith, N. Henrich, and J. Wolfe, "The acoustic impedance of the boehm flute: standard and some non-standard fingerings," *PROCEEDINGS-INSTITUTE OF ACOUSTICS*, vol. 19, pp. 315–320, 1997.

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

[67] S. J. Maclagan, *A dictionary for the modern flutist*. Scarecrow Press, 2009.

[68] Y. Han and K. Lee, "Detecting fingering of overblown flute sound using sparse feature learning," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2016, no. 1, pp. 1–10, 2016.

[69] C. Kereliuk, B. Scherrer, V. Verfaille, P. Depalle, and M. M. Wanderley, "Indirect acquisition of fingerings of harmonic notes on the flute," in *Proceedings of the International Computer Music Conference, ICMC '07, Copenhagen, Denmark*, vol. 1, 2007, pp. 263–266.

[70] V. Verfaille, P. Depalle, and M. M. Wanderley, "Detecting overblown flute fingerings from the residual noise spectrum," *The Journal of the Acoustical Society of America*, vol. 127, no. 1, pp. 534–541, 2010.

[71] K. W. Berger, "Some factors in the recognition of timbre," *The Journal of the Acoustical Society of America*, vol. 36, no. 10, pp. 1888–1891, 1964.

[72] A. Statnikov, L. Wang, and C. F. Aliferis, "A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification," *BMC bioinformatics*, vol. 9, no. 1, p. 1, 2008.

[73] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[74] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 161–168.

[75] D. C. Miller, "The influence of the material of wind-instruments on the tone quality," *Science*, pp. 161–171, 1909.

[76] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[77] J. J. Bosch, J. Janer, F. Fuhrmann, and P. Herrera, "A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals." in *ISMIR*, 2012, pp. 559–564.

[78] F. Fuhrmann, M. Haro, and P. Herrera, "Scalability, generality and temporal aspects in automatic recognition of predominant musical instruments in polyphonic music." in *ISMIR*, 2009, pp. 321–326.

[79] J. J. Burred, A. Robel, and T. Sikora, "Polyphonic musical instrument recognition based on a dynamic model of the spectral envelope," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2009, pp. 173–176.

[80] N. Chétry and M. Sandler, "Linear predictive models for musical instrument identification," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 5. IEEE, 2006, pp. V–V.

[81] Y. Han, S. Lee, J. Nam, and K. Lee, "Sparse feature learning for instrument identification: Effects of sampling and pooling methods," *The Journal of the Acoustical Society of America*, vol. 139, no. 5, pp. 2290–2298, 2016.

[82] S. Dieleman and B. Schrauwen, "Multiscale approaches to music audio feature learning," in *14th International Society for Music Information Retrieval Conference (ISMIR-2013)*. Pontifícia Universidade Católica do Paraná, 2013, pp. 116–121.

[83] P. Masri, "Computer modelling of sound for transformation and synthesis of musical signals." Ph.D. dissertation, University of Bristol, 1996.

[84] S. Dixon, "Onset detection revisited," in *Proceedings of the 9th International Conference on Digital Audio Effects*, vol. 120. Citeseer, 2006, pp. 133–137.

[85] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 111–118.

[86] A. Srinivasan, D. Sullivan, and I. Fujinaga, "Recognition of isolated instrument tones by conservatory students," in *Proc. International Conference on Music Perception and Cognition*, 2002, pp. 17–21.

[87] Y. Han, J. Kim, and K. 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, Jan 2017.

[88] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." in *INTERSPEECH*, vol. 2, 2010, p. 3.

[89] G. Mesnil, X. He, L. Deng, and Y. Bengio, "Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding." in *INTERSPEECH*, 2013, pp. 3771–3775.

[90] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu, "Recurrent neural networks for language understanding." in *INTERSPEECH*, 2013, pp. 2524–2528.

[91] T. Park and T. Lee, "Musical instrument sound classification with deep convolutional neural network using feature fusion approach," *arXiv preprint arXiv:1512.07370*, 2015.

[92] P. Li, J. Qian, and T. Wang, "Automatic instrument recognition in polyphonic music using convolutional neural networks," *arXiv preprint arXiv:1511.05520*, 2015.

[93] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer vision–ECCV 2014*. Springer, 2014, pp. 818–833.

[94] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[95] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[96] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[97] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from over-

fitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[98] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[99] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, and G. Wang, "Recent advances in convolutional neural networks," *arXiv preprint arXiv:1512.07108*, 2015.

[100] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.

[101] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, vol. 30, 2013, p. 1.

[102] K. He, X. Zhang, S. Ren, and J. 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.

[103] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[104] A. Krogh, J. Vedelsby *et al.*, "Neural network ensembles, cross validation, and active learning," *Advances in neural information processing systems*, vol. 7, pp. 231–238, 1995.

[105] F. Fuhrmann and P. Herrera, "Polyphonic instrument recognition for exploring semantic similarities in music," in *Proc. of 13th Int. Conference on Digital Audio Effects DAFx10*, 2010, pp. 1–8.

[106] N. Ono, K. Miyamoto, H. Kameoka, J. Le Roux, Y. Uchiyama, E. Tsunoo, T. Nishimoto, and S. Sagayama, "Harmonic and percussive sound separation and its application to mir-related tasks," in *Advances in music information retrieval*. Springer, 2010, pp. 213–236.

[107] M.-J. Kim, D.-K. Kang, and H. B. Kim, "Geometric mean based boosting algorithm with over-sampling to resolve data imbalance problem for bankruptcy prediction," *Expert Systems with Applications*, vol. 42, no. 3, pp. 1074–1082, 2015.

[108] R. Zhou and J. D. Reiss, "Music onset detection combining energy-based and pitch-based approaches," *Proc. MIREX Audio Onset Detection Contest*, 2007.

[109] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 2579-2605, p. 85, 2008.

[110] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," *arXiv preprint arXiv:1506.06579*, 2015.

[111] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

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

[113] Y. Han and K. Lee, "Acoustic scene classification using convolutional neural network and multiple-width frequency-delta data augmentation," *arXiv preprint arXiv:1607.02383*, 2016.

초록

음악 정보 분석 분야에서는 전통적으로 오디오에서 음고 및 음의 거친 정도와 밝기 등 오디오의 여러 특징들을 추출하기 위해서 규칙기반 방법을 이용한다. 하지만, 기계 학습을 통한 특징 추출을 이용한 방법들이 성능의 우수함 덕분에 최근 여러 분야에서 각광받고 있고, 특히 음악 정보 분석의 경우 여러 소리의 식별 또는 구분을 해야하는 경우에 유용하게 쓰인다. 본 논문의 궁극적인 목적은 음악 찾기, 검색, 추천, 교육 등에 널리 쓰일 수 있는 악기 식별 및 음색 분류 알고리즘의 최고 성능을 넘는 방법을 제시하는 것이다. 이를 위해 우리는 희소 특징 학습과 합성곱 신경망 알고리즘을 이용하여 입력 데이터에서 특징을 학습하는 네트워크 구조와 데이터 처리 프레임워크를 제안한다. 본 논문은 플룻 오버블로우 연주 기법 사용시의 운지법 검출, 단선율 음악에서의 악기 인식, 그리고 현실 환경의 다성 음악에서의 주악기 인식 등 크게 세가지 음악 정보 분석 분야에 특징 학습을 적용한 실험 결과를 포함하고 있다. 또한 이에 알맞는 데이터 프로세싱 기법 및 입력 프레임 추출, 주파수 척도, 활성화 통합 계층, 윈도우/홉 크기, 출력 집합화 방법, 신경망 학습 매개 변수를 찾기 위한 심도있는 실험을 포함한다.

주요어: 음악 정보 분석, 딥러닝, 합성곱 신경망, 희소 특징 학습, 악기 분류
학번: 2011-31242