# Deep Convolutional Neural Networks for Predominant Instrument Recognition in Polyphonic Music

Yoonchang Han, Jaehun Kim, and Kyogu Lee, *Senior Member, IEEE*

***Abstract*—Identifying musical instruments in polyphonic music recordings is a challenging but important problem in the field of music information retrieval. It enables music search by instrument, helps recognize musical genres, or can make music transcription easier and more accurate. In this paper, we present a convolutional neural network framework for predominant instrument recognition in real-world polyphonic music. We train our network from fixed-length music excerpts with a single-labeled predominant instrument and estimate an arbitrary number of predominant instruments from an audio signal with a variable length. To obtain the audio-excerpt-wise result, we aggregate multiple outputs from sliding windows over the test audio. In doing so, we investigated two different aggregation methods: one takes the class-wise average followed by normalization, and the other perform temporally local class-wise max-pooling on the output probability prior to averaging and normalization steps to minimize the effect of averaging process suppresses the activation of sporadically appearing instruments. In addition, we conducted extensive experiments on several important factors that affect the performance, including analysis window size, identification threshold, and activation functions for neural networks to find the optimal set of parameters. Our analysis on the instrument-wise performance found that the onset type is a critical factor for recall and precision of each instrument. Using a dataset of 10k audio excerpts from 11 instruments for evaluation, we found that convolutional neural networks are more robust than conventional methods that exploit spectral features and source separation with support vector machines. Experimental results showed that the proposed convolutional network architecture obtained an F1 measure of 0.619 for micro and 0.513 for macro, respectively, achieving 23.1% and 18.8% in performance improvement compared with the state-of-the-art algorithm.**

***Index Terms*—Convolutional neural networks, deep learning, instrument recognition, multi-layer neural network, music information retrieval.**

## I. Introduction

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 the extraction of information from 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 music information retrieval (MIR) field, it is highly desirable to know what instruments are used in the music. 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. In addition, the obtained instrument information can be used for various audio/music applications. For instance, it can be used for a tailored instrument-specific audio equalization and a music recommendation services. In addition, it can be used to enhance the performance of other MIR tasks. For example, knowing the number and type of the instrument can significantly improve the performance of audio source separation, automatic music transcription, and genre classification.

Instrument recognition can be performed in various forms. Hence, the term "instrument recognition" or "instrument identification" might indicate several different research topics. For instance, many of the related works focus on studio-recorded isolated notes. To name a few, Eronen used cepstral coefficients and temporal features to classify 30 orchestral instruments with several articulation styles and achieved a classification accuracy of 95% for instrument family level and about 81% for individual instruments [1]. 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 [2]. Yu *et al.* applied sparse coding on cepstrum with temporal sum-pooling and achieved an F-measure of about 96% for classifying

Manuscript received May 17, 2016; revised September 27, 2016 and November 7, 2016; accepted November 16, 2016. Date of publication November 23, 2016; date of current version December 16, 2016. This work was supported in part by the Ministry of Science, ICT and Future Planning (MSIP), South Korea, under the Information Technology Research Center support program (IITP-2016-H8501-16-1016) supervised by the Institute for Information & Communications Technology Promotion and in part by the National Research Foundation of Korea grant funded by the MSIP (NRF-2014R1A2A2A04002619). The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Matthew E. P. Davies.

Y. Han is with the Music and Audio Research Group, Graduate School of Convergence Science and Technology, Seoul National University, Seoul 08826, South Korea (e-mail: yoonchanghan@snu.ac.kr).

J. Kim was with the Seoul National University, Seoul 08826, South Korea. He is now with the Multimedia Computing Group, Delft University of Technology, Delft 2628 CD, The Netherlands (e-mail: J.H.Kim@tudelft.nl).

K. Lee is with the Music and Audio Research Group, Graduate School of Convergence Science and Technology, Seoul National University, Seoul 08826, South Korea, and also with the Advanced Institutes of Convergence Technology, Suwon 16229, South Korea (e-mail: kglee@snu.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASLP.2016.2632307

50 instruments [3]. They also reported their classification result on a multi-source database, which was about 66%.

Some previous works such as Krishna and Sreenivas [4] 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. In addition, Essid *et al.* [5] reported that a solo phrase classification using MFCCs and GMM along with principal components analysis (PCA) achieved an overall recognition accuracy of about 67% for five instruments.

More recent works deal with polyphonic sound, which is closer to real-world music compare 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.* [6] 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.* [7] 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 quartet. Duan *et al.* [8] 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 [9], was around 37% for two polyphony notes and 25% for six polyphony notes.

As shown above, most of the previous works focus on the identification of the instrument sounds in clean solo tones or phrases. More recent researches attempt to solve instrument identification in a polyphonic 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 paper, we investigate a method for predominant instrument recognition in professionally produced Western music recordings. We utilize convolutional neural networks (ConvNets) to learn the spectral characteristics of the music recordings with 11 musical 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 the paper is organized as follows. In Section II, 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 IV, 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 V, the Results section, with an analysis of the effects of analysis window size, aggregation strategy, activation function, and identification threshold. We also present an 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 VI.

## II. 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 [10]. 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 [11].

There are many variants and modified architectures of deep learning, depending on the target task. Especially, recurrent neural networks 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 [12] and spoken language understanding [13], [14].

On the other hand, ConvNet is useful for data with local groups of values that are highly correlated, forming distinctive local characteristics that might appear at different parts of the array [10]. It is one of the most popular approaches recently in the image processing area such as handwritten digit recognition [15]–[17] for the MNIST dataset and image tagging [18], [19] 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 [11], and also for speech-recognition tasks [20].

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 as in the image. ConvNets are usually composed of many convolutional layers, and inserting a pooling layer between convolutional layers allows the network to work at different time scales and introduces translation invariance with robustness against local distortions. 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 [21].

Hence, 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 [22], automatic chord recognition [23], [24], and music structure/boundary analysis [25], [26].

An attempt to apply ConvNets for musical instrument identification can be found in the recent report from Park *et al.* [27] and Li *et al.* [28], 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 [27] 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 [28] 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 [29], [30].

## III. System Architecture

### A. 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 [10]. Although it aims to learn a feature representation "automatically", appropriate preprocessing of input data is a crucial factor generating a good feature representation.

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, which is sufficient to cover most of the harmonics generated by musical instruments while removing noises possibly included in the frequencies above this range. Secondly, all audios are normalized by dividing the time-domain signal with its maximum value, and then it 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.* [31] and Hamel *et al.* [21], 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.

### B. Network Architecture

ConvNets can be seen as a combination of feature extractor and the classifier. Our ConvNet architecture is inspired by a popular AlexNet [18] and VGGNet [32] structure, which are very deep architecture using repeated several convolution layers followed by max-pooling, as shown in Fig. 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 [33] and Sermanet *et al.* [34], 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 $3 \times 3$ receptive field, with a fixed stride size of 1, and spatial abstraction is done by max-pooling with a size of $3 \times 3$ and a stride size of 1. In Table I, we illustrate the detailed ConvNet architecture with the input size in each layer and parameter values except the zero-padding process. The input for each convolution layer is zero-padded with $1 \times 1$ to preserve the spatial resolution regardless of input window size, and we increase the number of filters for the convolution layer by a factor of 2 after every two convolution layers, starting from 32 up to 256.

The last max-pooling layer prior to fully connected layer is a global max-pooling layer which takes a maximum value from all neurons in each channel. 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 [35]. However, our empirical

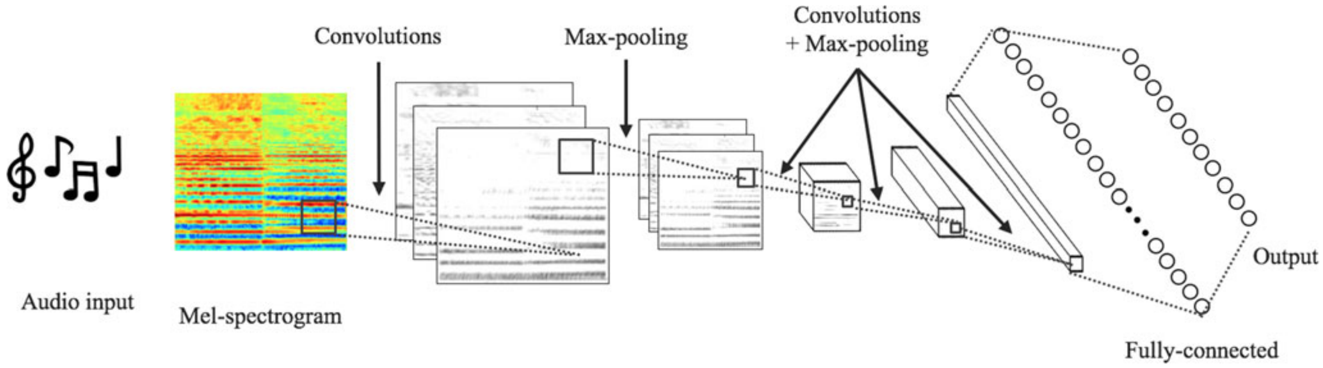


Fig. 1. Schematic of the proposed ConvNet containing 4 times repeated double convolution layers followed by max-pooling. The last max-pooling layer performs global max-pooling, then it is fed to a fully connected layer followed by 11 sigmoid outputs.

TABLE I
PROPOSED CONVNET STRUCTURE

| 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 |

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.

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.

It is common to use a softmax function for the classification layer when there is only one target label [18], [32], [35], but our system should be able to handle multiple instruments present at the same time. Hence, we used a sigmoid function for the classification layer at the end of the network.

### C. Training Configuration

The training was done by optimizing the categorical cross-entropy between predictions and targets. We used Adam [36] 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 [37]. Dropout rate after a fully connected layer was set to 0.5 because a fully connected layer easily suffers from overfitting [38].

In addition, we conducted an experiment with various time resolutions to find the optimal analysis size. As our training data were a fixed 3.0 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 training 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 [39].

### D. 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 using a hyperbolic tangent (tanh) or sigmoid function. However, non-saturating nonlinearities such as the rectified linear unit (ReLU) allow much faster learning than these saturating nonlinearities, particularly for models that are trained on large datasets [18]. A number of recent works have shown that the performance of ReLU is better than that of sigmoid and tanh activation [40] and most of the modern studies on ConvNets use ReLU to model the output of the neurons [28], [32]–[34].

ReLU was first introduced by Nair and Hinton in their work on restricted Boltzmann machines [41]. The ReLU activation function is defined as

$$y_i = \max(0, z_i) \tag{1}$$

where $z_i$ is the input of the $i$th 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. Leaky-ReLU (LReLU), introduced by Mass *et al.* [42], compresses the negative part rather than make it all zero, which might cause some initially inactive units to remain inactive. It is defined as

$$y_i = \begin{cases} z_i & z_i \geq 0 \\ \alpha z_i & z_i < 0 \end{cases} \tag{2}$$

where $\alpha$ is a parameter between 0 and 1 to give a small gradient in the negative part. Another modified version of ReLU, parametric ReLU (PReLU), is introduced by He *et al.* [43]. It is basically similar to LReLU, but it automatically learns the parameter for the negative gradient which is an input parameter for LReLU. It is defined as

$$y_i = \begin{cases} z_i & z_i \geq 0 \\ \alpha_i z_i & z_i < 0 \end{cases} \tag{3}$$

where $\alpha_i$ is the learned parameters for the $i$th 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.* [44] 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 $\alpha$. 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 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 [44].

## IV. Evaluation

### A. 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.* [45] 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 II with the objective of extracting music excerpts that contain a continuous presence of a single predominant instrument.

TABLE II
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 |

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.

### B. 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

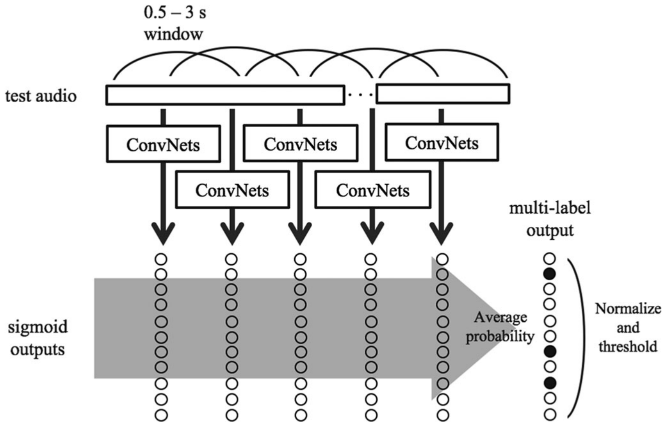


Fig. 2. Schematic of obtaining a multi-label output from a test audio signal. Input audio was analyzed with sliding window, and these multiple sigmoid outputs were aggregated using two different strategies, *S1* and *S2*, to estimate the predominant instrument for the testing audio excerpt.

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. Sliding window was used 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 accompaniments 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 Section IV-C.

We used values between 0.2 and 0.8 as an identification threshold $\theta$ 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 Fig. 2.

### *C. Performance Evaluation*

Following the evaluation method widely used in the instrument recognition task [45], [46], we computed the precision and recall, which are defined as:

$$P_l = \frac{tp_l}{tp_l + fp_l} \tag{4}$$

$$R_l = \frac{tp_l}{tp_l + fn_l} \tag{5}$$

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:

$$F1_l = \frac{2P_l R_l}{P_l + R_l} \tag{6}$$

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:

$$P_{\text{micro}} = \frac{\sum_{l=1}^{L} tp_l}{\sum_{l=1}^{L} tp_l + fp_l} \tag{7}$$

$$R_{\text{micro}} = \frac{\sum_{l=1}^{L} tp_l}{\sum_{l=1}^{L} tp_l + fn_l} \tag{8}$$

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 of all classes. Macro precision and recall are defined as:

$$P_{\text{macro}} = \frac{1}{|L|}\sum_{l=1}^{L} P_l \tag{9}$$

$$R_{\text{macro}} = \frac{1}{|L|}\sum_{l=1}^{L} R_l \tag{10}$$

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

$$F1_{\text{micro}} = \frac{2P_{\text{micro}}R_{\text{micro}}}{P_{\text{micro}} + R_{\text{micro}}} \tag{11}$$

$$F1_{\text{macro}} = \frac{2P_{\text{macro}}R_{\text{macro}}}{P_{\text{macro}} + R_{\text{macro}}} \tag{12}$$

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

## V. Results

In this section, we first demonstrate the result with various analysis resolution by varying the window size. Then, we observe the effect of adding max-pooling on the output probabilities, also illustrate the effect of identification threshold and activation functions. At the end of this section, we compare our

TABLE III
EXPERIMENT VARIABLES FOR THE ACTIVATION FUNCTION, SIZE OF THE ANALYSIS WINDOW, AGGREGATION STRATEGY, AND IDENTIFICATION THRESHOLD

| Variables | |
|---|---|
| analysis win. size | 0.5 s, **1.0 s**, 1.5 s, 3.0 s |
| $\theta$ (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 |

Default settings are indicated in bold.

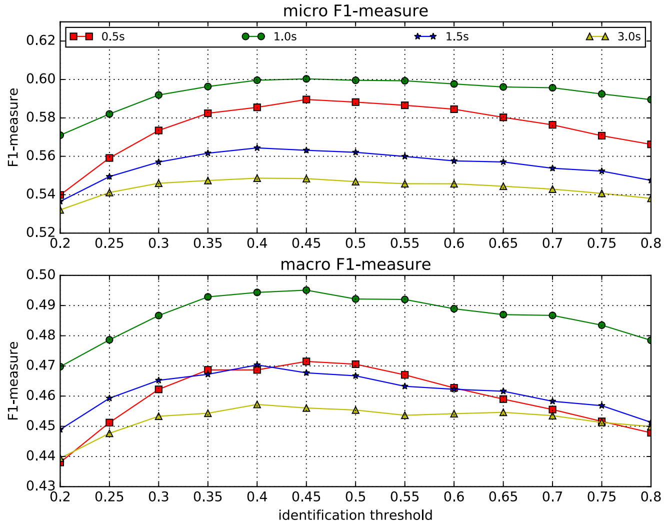


Fig. 3. Micro and macro $F1$ measure of an analysis window size of 0.5, 1.0, 1.5, and 3.0 s according to the identification threshold.

result to existing algorithms with optimal parameter settings found through the experiment.

We used ReLU for the activation function, 1.0 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 III and the effect of each variable is separately demonstrated in following sections.

### A. 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. Fig. 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.

However, shortening the analysis frame down to 0.5 s decreased the overall 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.

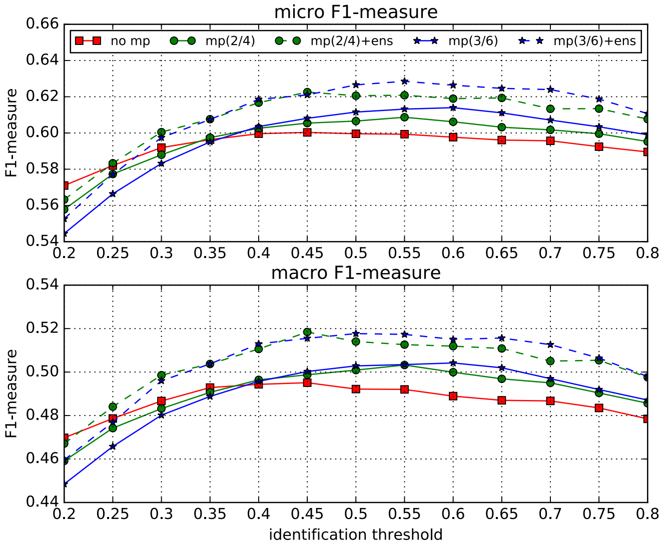


Fig. 4. Micro and macro $F1$ measure of without max-pooling, max-pooling with 4 and 6 frame, and mean model ensemble cases according to the identification threshold.

### B. Effect of Max-pooling and Model Ensemble

The class-wise max-pooling is performed prior to output averaging and normalization process by taking the maximum value for each class in the sliding window as explained in Section IV-B. 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 a hop size of half window size for the experiment. We experimented with various identification thresholds as in the analysis window size comparison experiment, because the optimal threshold value might change 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 Fig. 4. Optimal identification threshold was about 0.45 for the case without max-pooling, but it is increased to 0.55. It demonstrates 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 [47]. 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 4, it was possible to obtain further performance improvements via the use of a model

TABLE IV
INSTRUMENT RECOGNITION PERFORMANCE OF THE PROPOSED CONVNET WITH VARIOUS ACTIVATION FUNCTIONS

| 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 ($\alpha = 0.01$) | 0.651 | 0.611 | 0.631 | 0.524 | 0.550 | 0.524 |
| LReLU ($\alpha = 0.33$) | 0.658 | 0.567 | 0.609 | 0.545 | 0.509 | 0.508 |

Note that these results are from ensemble 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.

### C. 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 IV. However, unlike the results presented in the ImageNet classification work from He *et al.* [43] and empirical evaluation work on ConvNet activation function from Xu *et al.* [44], PRelu and LReLU did not show visible improvement from normal ReLU, but just showed nearly matched or slightly decreased performance.

To observe the 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 consequence, 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 does not provide statistically significant improvement for this task, unlike for computer vision tasks. Following this result, we decided to use normal ReLU as an activation function for our final model.

### D. 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 $\theta$, 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. Using the best settings we found from experiments, the proposed system 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 cover a wide variability in recording and production styles, also from various decades.

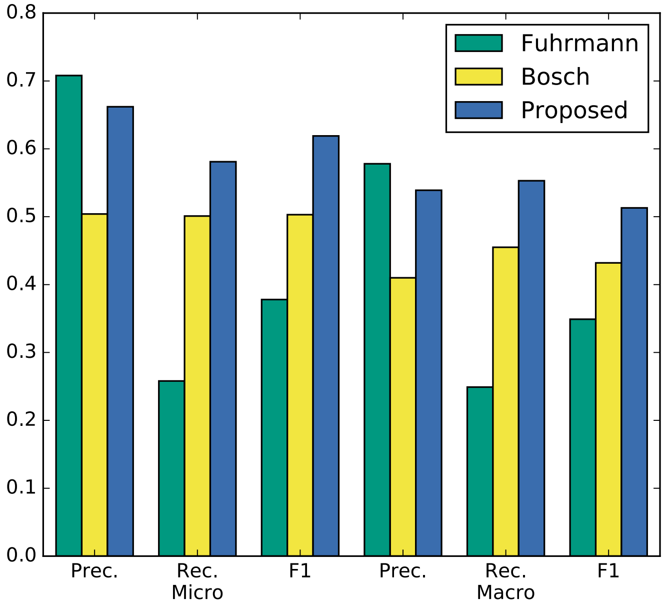


Fig. 5. Performance comparison of the predominant instrument recognition algorithm from Fuhrmann and Herrra [46], Bosch *et al.* [45], and our proposed ConvNet. Note that the result of proposed algorithm used half of the original test set, because another half was used for the development.

The existing algorithm from Fuhrmann and Herrera [46] used typical hand-made timbral audio features with their frame-wise mean and variance statistics to train SVMs, and Bosch *et al.* [45] improved this algorithm with source separation called FASST (Flexible Audio Source Separation Framework) [48] in a preprocessing step.

As shown in Fig. 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 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.

### E. 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 Fig. 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

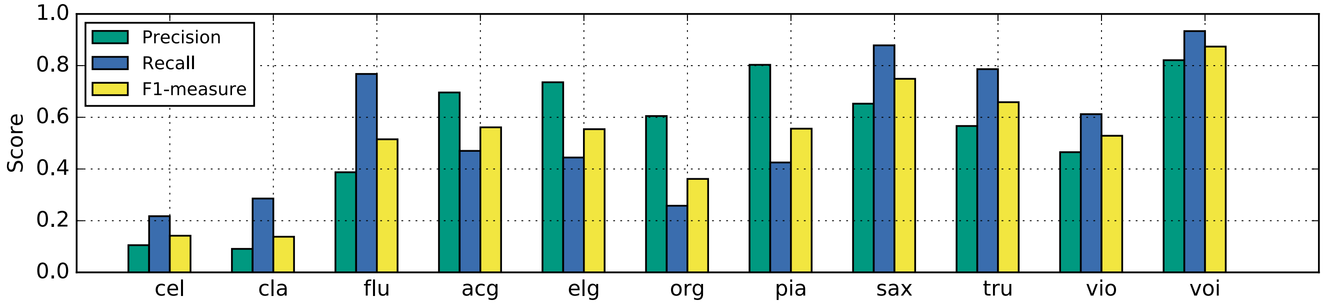


Fig. 6. Precision, recall, and F1-measure of each instrument with the optimal parameters.

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.

Apart from the issue related to the number of testing audio, 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. 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 instruments 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 [49]. 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.

### *F. 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 to the music with the 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 was 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 Fig. 8. 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 huge 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 the sounds of several instruments are overlapped. Also, it seems the aggregation process makes difference on the final performance. Note that because we used only half of the training set for this analysis, the performance of the actual model used for the multiple predominant instruments would be better than the result demonstrated in this section to some extent.

### *G. 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

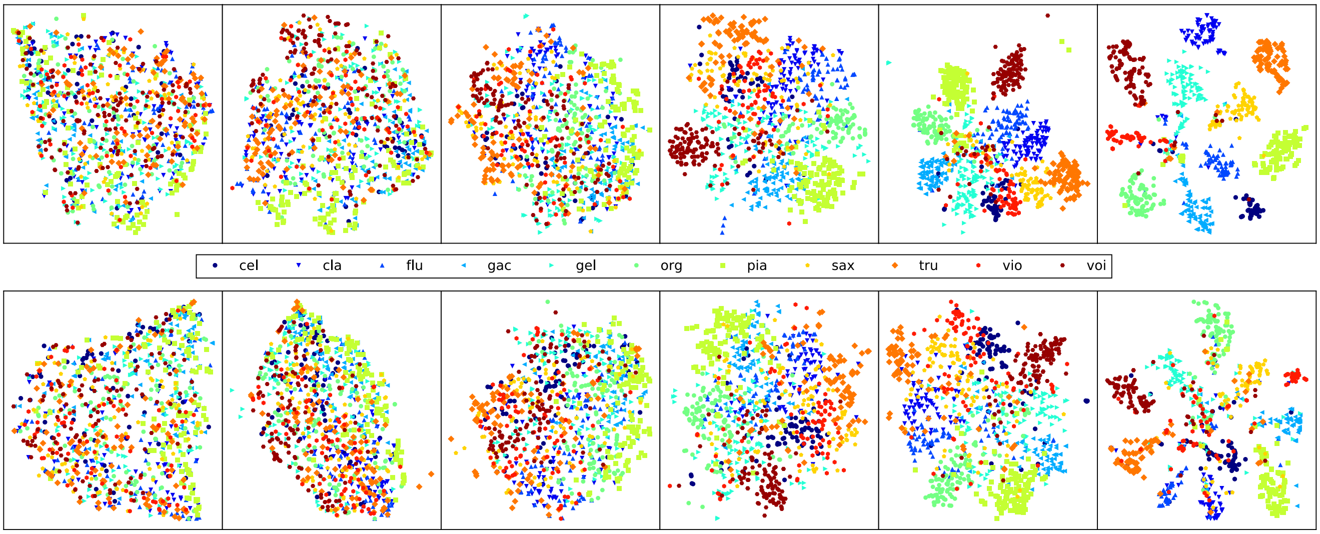
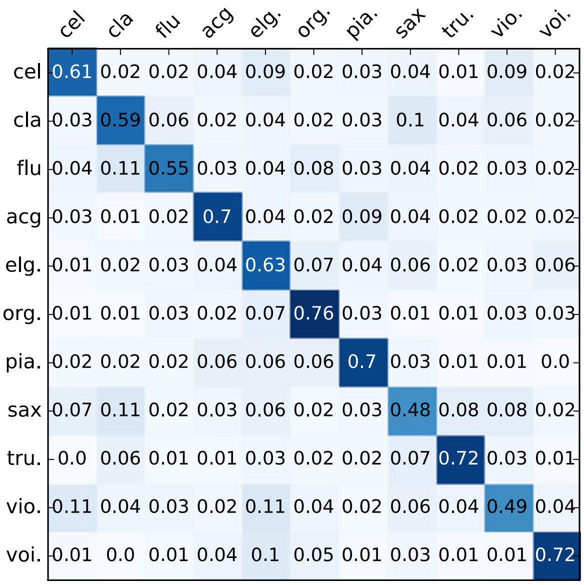


Fig. 7. Visualization of the t-SNE clustering result. It represents the clustering result for each intermediate state of the proposed model. From left to right, the first four plots are the clustering result of the activation at the end of each convolutional block, and the last two plots are the clustering result of the activations of the hidden dense layer and the final sigmoid output, respectively. The upper plots are drawn from the sample used in the training, and the lower plots are from the validation data samples.



| | cel | cla | flu | acg | elg. | org. | pia. | sax | tru. | vio. | voi. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| cel | 0.61 | 0.02 | 0.02 | 0.04 | 0.09 | 0.02 | 0.03 | 0.04 | 0.01 | 0.09 | 0.02 |
| cla | 0.03 | 0.59 | 0.06 | 0.02 | 0.04 | 0.02 | 0.03 | 0.1 | 0.04 | 0.06 | 0.02 |
| flu | 0.04 | 0.11 | 0.55 | 0.03 | 0.04 | 0.08 | 0.03 | 0.04 | 0.02 | 0.03 | 0.02 |
| acg | 0.03 | 0.01 | 0.02 | 0.7 | 0.04 | 0.02 | 0.09 | 0.04 | 0.02 | 0.02 | 0.02 |
| elg. | 0.01 | 0.02 | 0.03 | 0.04 | 0.63 | 0.07 | 0.04 | 0.06 | 0.02 | 0.03 | 0.06 |
| org. | 0.01 | 0.01 | 0.03 | 0.02 | 0.07 | 0.76 | 0.03 | 0.01 | 0.01 | 0.03 | 0.03 |
| pia. | 0.02 | 0.02 | 0.02 | 0.06 | 0.06 | 0.06 | 0.7 | 0.03 | 0.01 | 0.01 | 0.0 |
| sax | 0.07 | 0.11 | 0.02 | 0.03 | 0.06 | 0.02 | 0.03 | 0.48 | 0.08 | 0.08 | 0.02 |
| tru. | 0.0 | 0.06 | 0.01 | 0.01 | 0.03 | 0.02 | 0.02 | 0.07 | 0.72 | 0.03 | 0.01 |
| vio. | 0.11 | 0.04 | 0.03 | 0.02 | 0.11 | 0.04 | 0.02 | 0.06 | 0.04 | 0.49 | 0.04 |
| voi. | 0.01 | 0.0 | 0.01 | 0.04 | 0.1 | 0.05 | 0.01 | 0.03 | 0.01 | 0.01 | 0.72 |

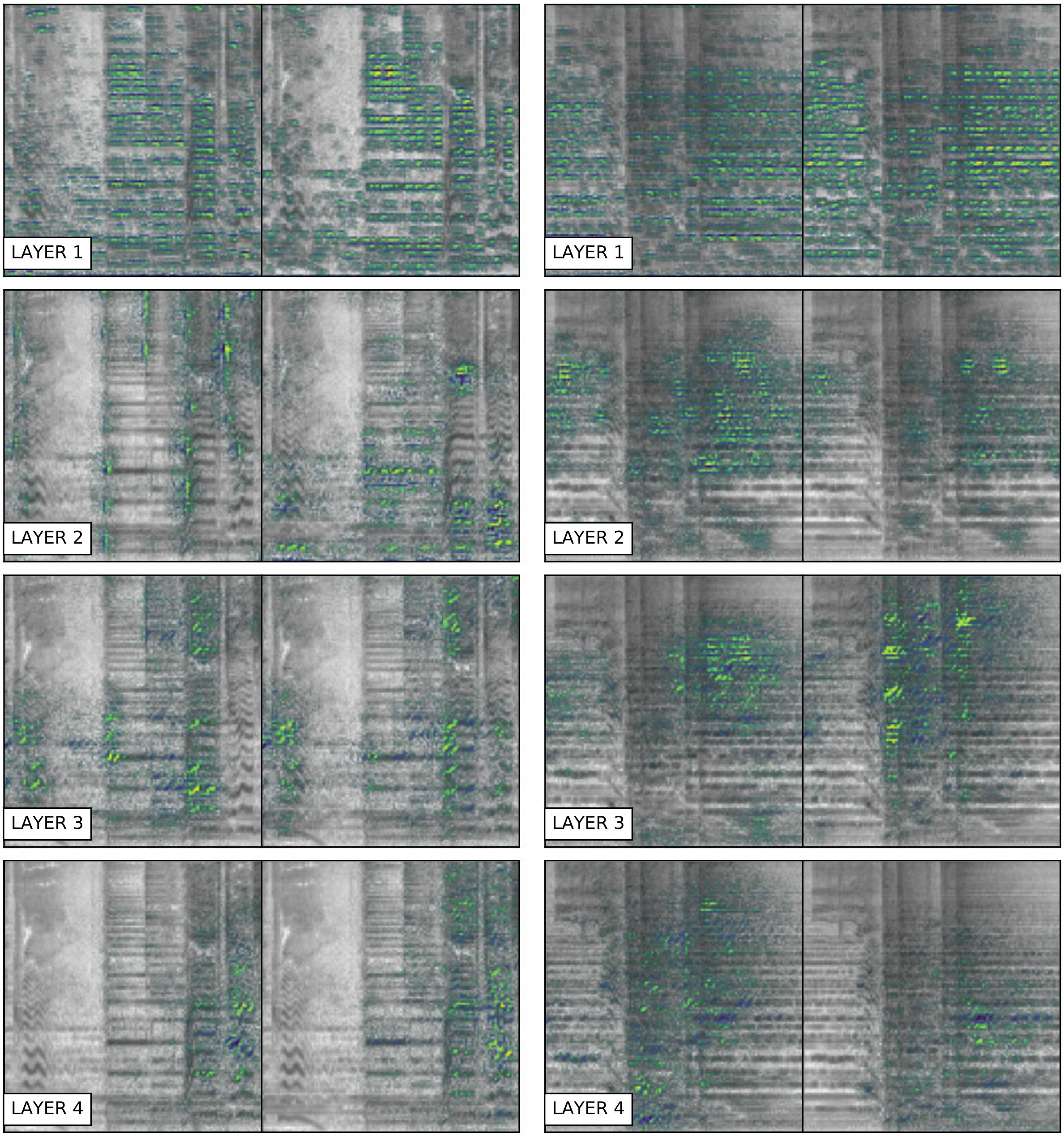
Fig. 8. Confusion matrix for single predominant instrument identification. X axis indicates predicted label and Y axis indicates true label.

embedding (t-SNE) [50] algorithm, which is a technique for dimensionality reduction of high-dimensional data. Second, we exploited the deconvolution [33], [51] 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 [50]. 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 activation 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 output 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 Fig. 7, 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 [33]. With this method, it is possible to reveal intuitively how each internal sub-function works within the entire



(a) (b)

Fig. 9. Mel-spectrogram of two input signals and their respective deconvoluted results. The left two columns and right two columns of the image, denoted as (a) and (b), respectively, were calculated from two independent music signals. Both signals were a 3 s polyphonic music segment that was randomly cropped from the original music. Moreover, both signals (a) and (b) consist mainly of the voice and the acoustic guitar sound. However, the dominant instrument of (a) is labeled as the voice, while (b) is labeled 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. For both target signals, the two highest activated units of each point were chosen and deconvoluted to be visualized. From left to right, images are arranged in order of decreasing absolute unit activation. The green and navy color indicate the positive and negative part of the result, respectively. The remaining area is where the magnitude of activation is relatively lower than in those regions. The range of activation result is normalized for the purpose of clear visualization.

deep convolutional network, which tends to be thought of as a “black box”.

By this process, the functionality of a subpart of the proposed model is explored. We generated deconvoluted images like those in Fig. 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 subpart 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 the top figures in Fig. 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 Fig. 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.

## VI. Conclusion

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 the 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 [52]. 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.

## References

[1] A. Eronen and A. Klapuri, "Musical instrument recognition using cepstral coefficients and temporal features," in *Proc. 2000 IEEE Int. Conf. Acoust., Speech Signal Process.*, 2000, vol. 2, pp. II753–II756.

[2] A. Diment, P. Rajan, T. Heittola, and T. Virtanen, "Modified group delay feature for musical instrument recognition," in *Proc. 10th Int. Symp. Comput. Music Multidiscip. Res.*, Marseille, France, 2013, pp. 431–438.

[3] L.-F. Yu, L. Su, and Y.-H. Yang, "Sparse cepstral codes and power scale for instrument identification," in *Proc. 2014 IEEE Int. Conf. Acoust., Speech Signal Process.*, 2014, pp. 7460–7464.

[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".

[4] A. Krishna and T. V. Sreenivas, “Music instrument recognition: From isolated notes to solo phrases,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2004, vol. 4, pp. iv-265–iv-268.

[5] S. Essid, G. Richard, and B. David, “Musical instrument recognition on solo performances,” in *Proc. 2004 12th Eur. Signal Process. Conf.*, 2004, pp. 1289–1292.

[6] T. Heittola, A. Klapuri, and T. Virtanen, “Musical instrument recognition in polyphonic audio using source-filter model for sound separation,” in *Proc. Int. Soc. Music Inf. Retrieval Conf.*, 2009, pp. 327–332.

[7] 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 J. Appl. Signal Process.*, vol. 2007, no. 1, pp. 155–155, 2007.

[8] Z. Duan, B. Pardo, and L. Daudet, “A novel cepstral representation for timbre modeling of sound sources in polyphonic mixtures,” in *Proc. 2014 IEEE Int. Conf. Acoust., Speech Signal Process.*, 2014, pp. 7495–7499.

[9] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, “RWC music database: Music genre database and musical instrument sound database,” in *Proc. Int. Soc. Music Inf. Retrieval Conf.*, 2003, vol. 3, pp. 229–230.

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

[11] L. Deng and D. Yu, “Deep learning: Methods and applications,” *Found. Trends Signal Process.*, vol. 7, no. 3–4, pp. 197–387, 2014.

[12] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, “Recurrent neural network based language model,” in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, 2010, vol. 2, pp. 1045–1048.

[13] G. Mesnil, X. He, L. Deng, and Y. Bengio, “Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding,” in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, 2013, pp. 3771–3775.

[14] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu, “Recurrent neural networks for language understanding,” in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, 2013, pp. 2524–2528.

[15] Y. LeCun *et al.*, “Learning algorithms for classification: A comparison on handwritten digit recognition,” *Neural Netw.: Stat. Mech. Perspect.*, vol. 261, pp. 261–276, 1995.

[16] A. Calderón, S. Roa, and J. Victorino, “Handwritten digit recognition using convolutional neural networks and Gabor filters,” in *Proc. Int. Congr. Comput. Intell*, 2003.

[17] X.-X. Niu and C. Y. Suen, “A novel hybrid CNN–SVM classifier for recognizing handwritten digits,” *Pattern Recog.*, vol. 45, no. 4, pp. 1318–1325, 2012.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[19] J. Ngiam, Z. Chen, D. Chia, P. W. Koh, Q. V. Le, and A. Y. Ng, “Tiled convolutional neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 1279–1287.

[20] G. Hinton *et al.*, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal Process. Mag.,* vol. 29, no. 6, pp. 82–97, Nov. 2012.

[21] P. Hamel, S. Lemieux, Y. Bengio, and D. Eck, “Temporal pooling and multiscale learning for automatic annotation and ranking of music audio,” in *Proc. Int. Soc. Music Inf. Retrieval Conf.*, 2011, pp. 729–734.

[22] J. Schluter and S. Bock, “Improved musical onset detection with convolutional neural networks,” in *Proc. 2014 IEEE Int. Conf. Acoust., Speech Signal Process.*, 2014, pp. 6979–6983.

[23] E. J. Humphrey and J. P. Bello, “Rethinking automatic chord recognition with convolutional neural networks,” in *Proc. 2012 11th Int. Conf. Mach. Learn. Appl.*, 2012, vol. 2, pp. 357–362.

[24] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, “Audio chord recognition with recurrent neural networks,” in *Proc. Int. Soc. Music Inf. Retrieval Conf.*, 2013, pp. 335–340.

[25] K. Ullrich, J. Schlüter, and T. Grill, “Boundary detection in music structure analysis using convolutional neural networks,” in *Proc. Int. Soc. Music Inf. Retrieval Conf.*, 2014, pp. 417–422.

[26] T. Grill and J. Schlüter, “Music boundary detection using neural networks on combined features and two-level annotations,” in *Proc. 16th Int. Soc. Music Inf. Retr. Conf.*, Malaga, Spain, 2015.

[27] T. Park and T. Lee, “Musical instrument sound classification with deep convolutional neural network using feature fusion approach,” arXiv:1512.07370, 2015.

[28] P. Li, J. Qian, and T. Wang, “Automatic instrument recognition in polyphonic music using convolutional neural networks,” arXiv:1511.05520, 2015.

[29] Y. Hoshen, R. J. Weiss, and K. W. Wilson, “Speech acoustic modeling from raw multichannel waveforms,” in *Proc. 2015 IEEE Int. Conf., Acoust., Speech Signal Process.*, 2015, pp. 4624–4628.

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

[31] J. Nam, J. Herrera, M. Slaney, and J. O. Smith, “Learning sparse feature representations for music annotation and retrieval,” in *Proc. Int. Soc. Music Inf. Retrieval Conf.*, 2012, pp. 565–570.

[32] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv:1409.1556, 2014.

[33] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 818–833.

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

[35] M. Lin, Q. Chen, and S. Yan, “Network in network,” arXiv:1312.4400, 2013.

[36] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv:1412.6980, 2014.

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[38] Y. Zhang, B. Zhou, H. Wu, and C. Wen, “2D fake fingerprint detection based on improved CNN and local descriptors for smart phone,” in *Proc. Chin. Conf. Biometric Recog.*, 2016, pp. 655–662.

[39] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proc. Int. Conf. Artif. Intell. Stat.*, 2010, pp. 249–256.

[40] J. Gu *et al.*, “Recent advances in convolutional neural networks,” arXiv:1512.07108, 2015.

[41] V. Nair and G. E. Hinton, “Rectified linear units improve restricted Boltzmann machines,” in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 807–814.

[42] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in *Proc. Int. Conf. Mach. Learn.*, 2013, vol. 30, p. 1.

[43] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.

[44] B. Xu, N. Wang, T. Chen, and M. Li, “Empirical evaluation of rectified activations in convolutional network,” arXiv:1505.00853, 2015.

[45] 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 *Proc. Int. Soc. Music Inf. Retrieval Conf.*, 2012, pp. 559–564.

[46] F. Fuhrmann and P. Herrera, “Polyphonic instrument recognition for exploring semantic similarities in music,” in *Proc. 13th Int. Conf. Digit. Audio Effects*, 2010, pp. 1–8.

[47] A. Krogh *et al.*, “Neural network ensembles, cross validation, and active learning,” *Adv. Neural Inf. Process. Syst.*, vol. 7, pp. 231–238, 1995.

[48] N. Ono *et al.*, “Harmonic and percussive sound separation and its application to MIR-related tasks,” in *Advances in Music Information Retrieval*. Berlin, Germany: Springer, 2010, pp. 213–236.

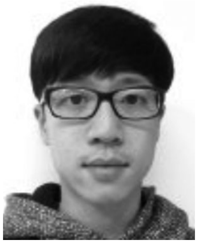
[49] R. Zhou and J. D. Reiss, “Music onset detection combining energy-based and pitch-based approaches,” in *Proc. MIREX Audio Onset Detect. Contest*, 2007.

[50] L. Van der Maaten and G. Hinton, “Visualizing data using t-SNE,” *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, 2008.

[51] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, “Understanding neural networks through deep visualization,” arXiv:1506.06579, 2015.

[52] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” arXiv:1412.6572, 2014.

**Yoonchang Han** was born in Seoul, South Korea, in 1986. He studied electronic engineering systems at King’s College London, U.K., from 2006 to 2009, and then moved to Queen Mary University of London, U.K., and received an M.Eng.(Hons.) degree in digital audio and music system engineering with First Class Honours in 2011. He is currently working toward the Ph.D. degree in digital contents and information studies at the Music and Audio Research Group, Seoul National University, Seoul, South Korea. His research interest includes developing deep learning techniques for automatic musical instrument recognition.



**Jaehun Kim** was born in Seoul, South Korea, in 1986. He received the B.A. degree in english literature and linguistics from the Seoul National University, Seoul, South Korea and the M.S. degree in the digital contents and information studies from the Music and Audio Research Group (MARG), Seoul National University, Seoul, South Korea. He is currently a Researcher in the MARG. His research interests include signal processing and machine learning techniques applied to music and audio.

**Kyogu Lee** received the Ph.D. degree in computer-based music theory and acoustics from the Stanford University, Stanford, CA, USA. He is an Associate Professor at the Seoul National University and leads the Music and Audio Research Group. His research interests include signal processing and machine learning techniques applied to music and audio.