SAMPLE-LEVEL DEEP CONVOLUTIONAL NEURAL NETWORKS FOR MUSIC AUTO-TAGGING USING RAW WAVEFORMS

Jongpil Lee **Jiyoung Park** **Keunhyoung Luke Kim** **Juhan Nam**
Korea Advanced Institute of Science and Technology (KAIST)
[richter, jypark527, dilu, juhannam]@kaist.ac.kr

ABSTRACT

Recently, the end-to-end approach that learns hierarchical representations from raw data using deep convolutional neural networks has been successfully explored in the image, text and speech domains. This approach was applied to musical signals as well but has been not fully explored yet. To this end, we propose sample-level deep convolutional neural networks which learn representations from very small grains of waveforms (e.g. 2 or 3 samples) beyond typical frame-level input representations. Our experiments show how deep architectures with sample-level filters improve the accuracy in music auto-tagging and they provide results comparable to previous state-of-the-art performances for the Magnatagatune dataset and Million Song Dataset. In addition, we visualize filters learned in a sample-level DCNN in each layer to identify hierarchically learned features and show that they are sensitive to log-scaled frequency along layer, such as mel-frequency spectrogram that is widely used in music classification systems.

1. INTRODUCTION

In music information retrieval (MIR) tasks, raw waveforms of music signals are generally converted to a time-frequency representation and used as input to the system. The majority of MIR systems use a log-scaled representation in frequency such as mel-spectrograms and constant-Q transforms and then compress the amplitude with a log scale. The time-frequency representations are often transformed further into more compact forms of audio features depending on the task. All of these processes are designed based on acoustic knowledge or engineering efforts.

Recent advances in deep learning, especially the development of deep convolutional neural networks (DCNN), made it possible to learn the entire hierarchical representations from the raw input data, thereby minimizing the input data processing by hands. This end-to-end hierarchical learning was attempted early in the image domain, particularly since the DCNN achieves break-through results in image classification [1]. These days, the method of stacking small filters (e.g. 3x3) is widely used after it has been found to be effective in learning more complex hierarchical filters while conserving receptive fields [2]. In the text domain, the language model typically consists of two steps: word embedding and word-level learning. While word embedding plays a very important role in language processing [3], it has limitations in that it is learned independently from the system. Recent work using CNNs that take character-level text as input showed that the end-to-end learning approach can yield comparable results to the word-level learning system [4, 5]. In the audio domain, learning from raw audio has been explored mainly in the automatic speech recognition task [6–10]. They reported that the performance can be similar to or even superior to that of the models using spectral-based features as input.

This end-to-end learning approach has been applied to music classification tasks as well [11, 12]. In particular, Dieleman and Schrauwen used raw waveforms as input of CNN models for music auto-tagging task and attempted to achieve comparable results to those using mel-spectrograms as input [11]. Unfortunately, they failed to do so and attributed the result to three reasons. First, their CNN models were not sufficiently expressive (e.g. a small number of layers and filters) to learn the complex structure of polyphonic music. Second, they could not find an appropriate non-linearity function that can replace the log-based amplitude compression in the spectrogram. Third, the bottom layer in the networks takes raw waveforms in frame-level which are typically several hundred samples long. The filters in the bottom layer should learn all possible phase variations of periodic waveforms which are likely to be prevalent in musical signals. The phase variations within a frame (i.e. time shift of periodic waveforms) are actually removed in the spectrogram.

In this paper, we address these issues with sample-level DCNN. What we mean by "sample-level" is that the filter size in the bottom layer may go down to several samples long. We assume that this small granularity is analogous to pixel-level in image or character-level in text. We show the effectiveness of the sample-level DCNN in music auto-tagging task by decreasing strides of the first convolutional layer from frame-level to sample-level and accordingly increasing the depth of layers. Our experiments show that the depth of architecture with sample-level filters is proportional to the accuracy and also the architecture achieves results comparable to previous state-of-the-art performances for the MagnaTagATune dataset and the Million Song Dataset. In addition, we visualize filters learned in the sample-level DCNN.

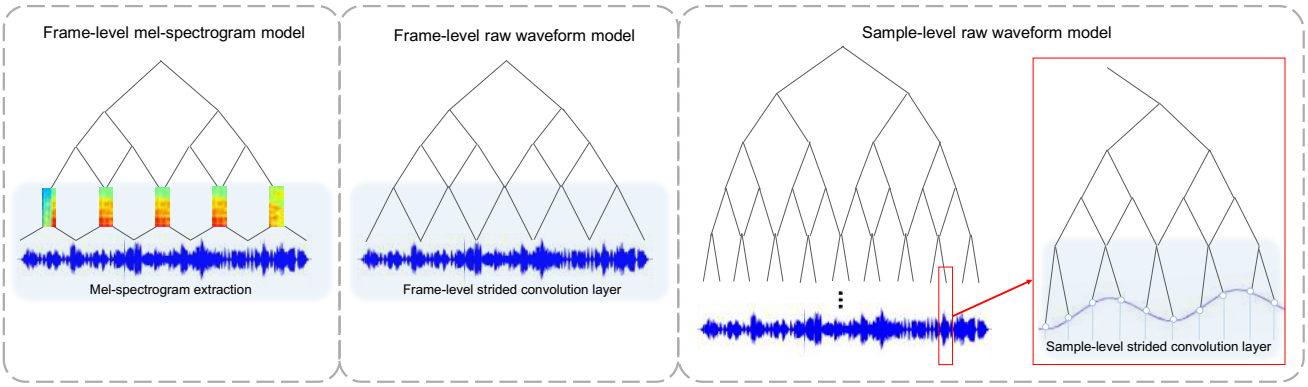

Figure 1. Simplified model comparison of frame-level approach using mel-spectrogram (left), frame-level approach using raw waveforms (middle) and sample-level approach using raw waveforms (right).

2. RELATED WORK

Since audio waveforms are one-dimensional data, previous work that takes waveforms as input used a CNN that consists of one-dimensional convolution and pooling stages. While the convolution operation and filter length in upper layers are usually similar to those used in the image domain, the bottom layer that takes waveform directly conducted a special operation called *strided convolution*, which takes a large filter length and strides it as much as the filter length (or the half). This frame-level approach is comparable to hopping windows with 100% or 50% hop size in a short-time Fourier transform. In many previous works, the stride and filter length of the first convolution layer was set to 10-20 ms (160-320 samples at 16 kHz audio) [8, 10–12].

In this paper, we reduce the filter length and stride of the first convolution layer to the sample-level, which can be as small as 2 samples. Accordingly, we increase the depth of layers in the CNN model. There are some works that use 0.6 ms (10 samples at 16 kHz audio) as a stride length [6, 7], but they used a CNN model only with three convolution layers, which is not sufficient to learn the complex structure of musical signals.

3. LEARNING MODELS

Figure 1 illustrates three CNN models in the music auto-tagging task we compare in our experiments. In this section, we describe the three models in detail.

3.1 Frame-level mel-spectrogram model

This is the most common CNN model used in music auto-tagging. Since the time-frequency representation is two dimensional data, previous work regarded it as either two-dimensional images or one-dimensional sequence of vectors [11, 13–15]. We only used one-dimensional(1D) CNN model for experimental comparisons in our work because the performance gap between 1D and 2D models is not significant and 1D model can be directly compared to models using raw waveforms.

3.2 Frame-level raw waveform model

In the frame-level raw waveform model, a strided convolution layer is added beneath the bottom layer of the frame-level mel-spectrogram model. The strided convolution layer is expected to learn a filter-bank representation that correspond to filter kernels in a time-frequency representation. In this model, once the raw waveforms pass through the first strided convolution layer, the output feature map has the same dimensions as the mel-spectrogram. This is because the stride, filter length, and number of filters of the first convolution layer correspond to the hop size, window size, and number of mel-bands in the mel-spectrogram, respectively. This configuration was used for music auto-tagging task in [11, 12] and so we used it as a baseline model.

3.3 Sample-level raw waveform model

As described in Section 1, the approach using the raw waveforms should be able to address log-scale amplitude compression and phase-invariance. Simply adding a strided convolution layer is not sufficient to overcome the problems. To improve this, we add multiple layers beneath the frame-level such that the first convolution layer can handle much smaller length of samples. For example, if the stride of the first convolution layer is reduced from 729 ($= 3^6$) to 243 ($= 3^5$), 3-size convolution layer and max-pooling layer are added to keep the output dimensions in the subsequent convolution layers unchanged. If we repeatedly reduce the stride of the first convolution layer this way, six convolution layers (five pairs of 3-size convolution and max-pooling layer following one 3-size strided convolution layer) will be added (we assume that the temporal dimensionality reduction occurs only through max-pooling and striding while zero-padding is used in convolution to preserve the size). We describe more details on the configuration strategy of sample-level CNN model in the following section.

3.4 Model Design

Since the length of an audio clip is variable in general, the following issues should be considered when configuring the temporal CNN architecture:

- Convolution filter length and sub-sampling length
- The temporal length of hidden layer activations on the last sub-sampling layer
- The segment length of audio that corresponds to the input size of the network

First, we attempted a very small filter length in convolutional layers and sub-sampling length, following the VGG net that uses filters of 3×3 size and max-pooling of 2×2 size [2]. Since we use one-dimensional convolution and sub-sampling for raw waveforms, however, the filter length and pooling length need to be varied. We thus constructed several DCNN models with different filter length and pooling length from 2 to 5, and verified the effect on music auto-tagging performance. As a sub-sampling method, max-pooling is generally used. Although sub-sampling using strided convolution has recently been proposed in a generative model [9], our preliminary test showed that max-pooling was superior to the stride sub-sampling method. In addition, to avoid exhausting model search, a pair of single convolution layer and max-pooling layer with the same size was used as a basic building module of the DCNN.

Second, the temporal length of hidden layer activations on the last sub-sampling layer reflects the temporal compression of the input audio by successive sub-sampling. We set the CNN models such that the temporal length of hidden layer activation is one. By building the models this way, we can significantly reduce the number of parameters between the last sub-sampling layer and the output layer. Also, we can examine the performance only by the depth of layers and the stride of first convolution layer.

Third, in music classification tasks, the input size of the network is an important parameter that determines the classification performance [16, 17]. In the mel-spectrogram model, one song is generally divided into small segments with 1 to 4 seconds. The segments are used as the input for training and the predictions over all segments in one song are averaged in testing. In the models that use raw waveform, the learning ability according to the segment size has been not reported yet and thus we need to examine different input sizes when we configure the CNN models.

Considering all of these issues, we construct m^n-DCNN models where m refers to the filter length and pooling length of intermediate convolution layer modules and n refers to the number of the modules (or depth). An example of m^n-DCNN models is shown in Table 1 where m is 3 and n is 9. According to the definition, the filter length and pooling length of the convolution layer are 3 other than the first strided convolution layer. If the hop size (stride length) of the first strided convolution layer is 3, the time-wise output dimension of the convolution layer becomes 19683 when the input of the network is 59049 samples. We call this "3^9 model with 19683 frames and 59049 samples as input".

4. EXPERIMENTAL SETUP

In this section, we introduce the datasets used in our experiments and describe experimental settings.

3^9 model, 19683 frames 59049 samples (2678 ms) as input			
layer	stride	output	# of params
conv 3-128	3	19683×128	512
conv 3-128	1	19683×128	49280
maxpool 3	3	6561×128	
conv 3-128	1	6561×128	49280
maxpool 3	3	2187×128	
conv 3-256	1	2187×256	98560
maxpool 3	3	729×256	
conv 3-256	1	729×256	196864
maxpool 3	3	243×256	
conv 3-256	1	243×256	196864
maxpool 3	3	81×256	
conv 3-256	1	81×256	196864
maxpool 3	3	27×256	
conv 3-256	1	27×256	196864
maxpool 3	3	9×256	
conv 3-256	1	9×256	196864
maxpool 3	3	3×256	
conv 3-512	1	3×512	393728
maxpool 3	3	1×512	
conv 1-512	1	1×512	262656
dropout 0.5	–	1×512	
sigmoid	–	50	25650
Total params			1.9×10^6

Table 1. Sample-level CNN configuration. For example, in the layer column, the first 3 of "conv 3-128" is the filter length, 128 is the number of filters, and 3 of "maxpool 3" is the pooling length.

4.1 Datasets

We evaluate the proposed model on two datasets, MagnaTagATune dataset (MTAT) [18] and Million Song Dataset (MSD) annotated with the Last.FM tags [19]. We primarily examined the proposed model on MTAT and then verified the effectiveness of our model on MSD which is much larger than MTAT [1]. We filtered out the tags and used most frequently labeled 50 tags in both datasets, following the previous work [11], [14, 15] [2]. Also, all songs in the two datasets were trimmed to 29.1 second long and resampled to 22050 Hz as needed. We used AUC (Area Under Receiver Operating Characteristic) as a primary evaluation metric for music auto-tagging.

[1] MTAT contains 170 hours long audio and MSD contains 1955 hours long audio in total

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

2^n models

model with 16384 samples (743 ms) as input					model with 32768 samples (1486 ms) as input				
model	n	layer	filter length & stride	AUC	model	n	layer	filter length & stride	AUC
64 frames	6	1+6+1	256	0.8839	128 frames	7	1+7+1	256	0.8834
128 frames	7	1+7+1	128	0.8899	256 frames	8	1+8+1	128	0.8872
256 frames	8	1+8+1	64	0.8968	512 frames	9	1+9+1	64	0.8980
512 frames	9	1+9+1	32	0.8994	1024 frames	10	1+10+1	32	0.8988
1024 frames	10	1+10+1	16	0.9011	2048 frames	11	1+11+1	16	0.9017
2048 frames	11	1+11+1	8	0.9031	4096 frames	12	1+12+1	8	0.9031
4096 frames	12	1+12+1	4	0.9036	8192 frames	13	1+13+1	4	0.9039
8192 frames	13	1+13+1	2	0.9032	16384 frames	14	1+14+1	2	0.9040

3^n models

model with 19683 samples (893 ms) as input					model with 59049 samples (2678 ms) as input				
model	n	layer	filter length & stride	AUC	model	n	layer	filter length & stride	AUC
27 frames	3	1+3+1	729	0.8655	81 frames	4	1+4+1	729	0.8655
81 frames	4	1+4+1	243	0.8753	243 frames	5	1+5+1	243	0.8823
243 frames	5	1+5+1	81	0.8961	729 frames	6	1+6+1	81	0.8936
729 frames	6	1+6+1	27	0.9012	2187 frames	7	1+7+1	27	0.9002
2187 frames	7	1+7+1	9	0.9033	6561 frames	8	1+8+1	9	0.9030
6561 frames	8	1+8+1	3	0.9039	19683 frames	9	1+9+1	3	**0.9055**

4^n models

model with 16384 samples (743 ms) as input					model with 65536 samples (2972 ms) as input				
model	n	layer	filter length & stride	AUC	model	n	layer	filter length & stride	AUC
64 frames	3	1+3+1	256	0.8828	256 frames	4	1+4+1	256	0.8813
256 frames	4	1+4+1	64	0.8968	1024 frames	5	1+5+1	64	0.8950
1024 frames	5	1+5+1	16	0.9010	4096 frames	6	1+6+1	16	0.9001
4096 frames	6	1+6+1	4	0.9021	16384 frames	7	1+7+1	4	0.9026

5^n models

model with 15625 samples (709 ms) as input					model with 78125 samples (3543 ms) as input				
model	n	layer	filter length & stride	AUC	model	n	layer	filter length & stride	AUC
125 frames	3	1+3+1	125	0.8901	625 frames	4	1+4+1	125	0.8870
625 frames	4	1+4+1	25	0.9005	3125 frames	5	1+5+1	25	0.9004
3125 frames	5	1+5+1	5	0.9024	15625 frames	6	1+6+1	5	0.9041

Table 2. Comparison of various m^n-DCNN models with different input sizes. m refers to the filter length and pooling length of intermediate convolution layer modules and n refers to the number of the modules. Filter length & stride indicates the value of the first convolution layer. In the layer column, the first digit '1' of 1+n+1 is the strided convolution layer, and the last digit '1' is convolution layer which actually works as a fully-connected layer.

4.2 Optimization

We used sigmoid activation for the output layer and binary cross entropy loss as the objective function to optimize. For every convolution layer, we used batch normalization [20] and ReLU activation. We should note that, in our experiments, batch normalization plays a vital role in training the deep models that takes raw waveforms. We applied dropout of 0.5 to the output of the last convolution layer and minimized the objective function using stochastic gradient descent with 0.9 Nesterov momentum. The learning rate was initially set to 0.01 and decreased by a factor of 5 when the validation loss did not decrease more than 3 epochs. A total decrease of 4 times, the learning rate of the last training was 0.000016. Also, we used batch size of 23 for MTAT and 50 for MSD, respectively. In the mel-spectrogram model, we conducted the input normalization simply by dividing the standard deviation after subtracting mean value of entire input data. On the other hand, we did not perform the input normalization for raw waveforms.

5. RESULTS

In this section, we examine the proposed models and compare them to previous state-of-the-art results.

5.1 m^n-DCNN models

Table 2 shows the evaluation results for the m^n-DCNN models on MTAT for different input sizes, number of layers, filter length and stride of the first convolution layer. As

3^n models, 59049 samples as input	n	window (filter length)	hop (stride)	AUC
Frame-level (mel-spectrogram)	4	729	729	0.9000
	5	729	243	0.9005
	5	243	243	0.9047
	6	243	81	**0.9059**
	6	81	81	0.9025
Frame-level (raw waveforms)	4	729	729	0.8655
	5	729	243	0.8742
	5	243	243	0.8823
	6	243	81	0.8906
	6	81	81	0.8936
Sample-level (raw waveforms)	7	27	27	0.9002
	8	9	9	0.9030
	9	3	3	**0.9055**

Table 3. Comparison of three CNN models with different window (filter length) and hop (stride) sizes. n represents the number of intermediate convolution and max-pooling layer modules, thus 3^n times hop (stride) size of each model is equal to the number of input samples.

input type	model	MTAT	MSD
Frame-level (mel-spectrogram)	Persistent CNN [21]	0.9013	-
	2D CNN [14]	0.894	0.851
	CRNN [15]	-	0.862
	Proposed DCNN	**0.9059**	-
Frame-level (raw waveforms)	1D CNN [11]	0.8487	-
Sample-level (raw waveforms)	Proposed DCNN	**0.9055**	**0.8812**

Table 4. Comparison of our works to prior state-of-the-arts

described in Section 3.4, m refers to the filter length and pooling length of intermediate convolution layer modules and n refers to the number of the modules. In Table 2, we can first find that the accuracy is proportional to n for most m. Increasing n given m and input size indicates that the filter length and stride of the first convolution layer become closer to the sample-level (e.g. 2 or 3 size). When the first layer reaches the small granularity, the architecture is seen as a model constructed with the same filter length and sub-sampling length in all convolution layers as depicted in Table 1. The best results were obtained when m was 3 and n was 9. Interestingly, the length of 3 corresponds to the 3-size spatial filters in the VGG net [2]. In addition, we can see that 1-3 seconds of audio as an input length to the network is a reasonable choice in the raw waveform model as in the mel-spectrogram model.

5.2 Mel-spectrogram and raw waveforms

Considering that the output size of the first convolution layer in the raw waveform models is equivalent to the mel-spectrogram size, we further validate the effectiveness of the proposed sample-level architecture by performing experiments presented in Table 3. The models used in the experiments follow the configuration strategy described in Section 3.4. In the mel-spectrogram experiments, 128 mel-bands are used to match up to the number of filters in the first convolution layer of the raw waveform model. FFT size was set to 729 in all comparisons and the magnitude compression is applied with a nonlinear curve, $\log(1 + C|A|)$ where A is the magnitude and C is set to 10.

The results in Table 3 show that the sample-level raw waveform model achieves results comparable to the frame-level mel-spectrogram model. Specifically, we found that using a smaller hop size (81 samples $\approx$ 4 ms) worked better than those of conventional approaches (about 20 ms or so) in the frame-level mel-spectrogram model. However, if the hop size is less than 4 ms, the performance degraded. An interesting finding from the result of the frame-level raw waveform model is that when the filter length is larger than the stride, the accuracy is slightly lower than the models with the same filter length and stride. We interpret that this result is due to the learning ability of the phase variance. As the filter length decreases, the extent of phase variance that the filters should learn is reduced.

5.3 MSD result and the number of filters

We investigate the capacity of our sample-level architecture even further by evaluating the performance on MSD that is ten times larger than MTAT. The result is shown in Table 4. While training the network on MSD, the number of filters in the convolution layers has been shown to affect the performance. According to our preliminary test results, increasing the number of filters from 16 to 512 along the layers was sufficient for MTAT. However, the test on MSD shows that increasing the number of filters in the first convolution layer improves the performance. Therefore, we increased the number of filters in the first convolution layer from 16 to 128.

5.4 Comparison to state-of-the-arts

In Table 4, we show the performance of the proposed architecture to previous state-of-the-arts on MTAT and MSD. They show that our proposed sample-level architecture is highly effective compared to them.

5.5 Visualization of learned filters

The technique of visualizing the filters learned at each layer allows better understanding of representation learning in the hierarchical network. However, many previous works in music domain are limited to visualizing learned filters only on the first convolution layer [11, 12].

The gradient ascent method has been proposed for filter visualization [22] and this technique has provided deeper understanding of what convolutional neural networks learn from images [23, 24]. We applied the technique to our DCNN to observe how each layer hears the raw waveforms. The gradient ascent method is as follows. First, we generate random noise and back-propagate the errors in the network. The loss is set to the target filter activation. Then,

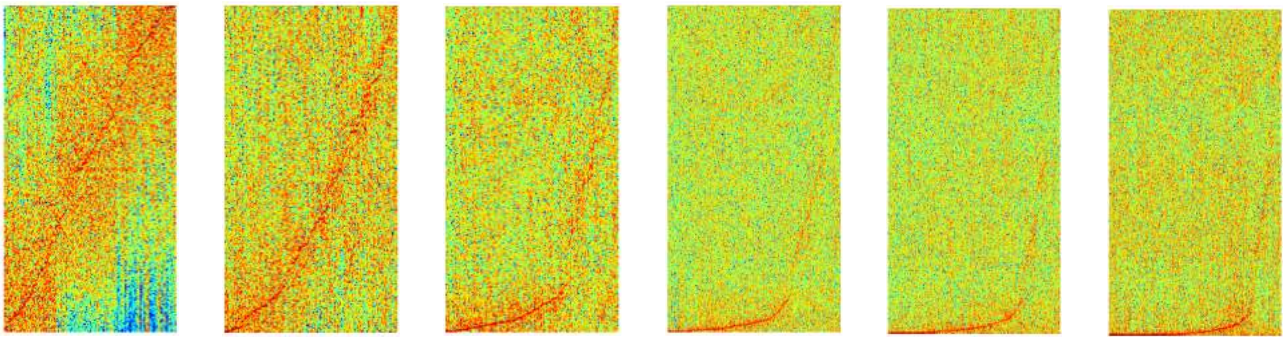

Figure 2. Spectrum of the filters in the sample-level convolution layers which are sorted by the frequency of the peak magnitude. The x-axis represents the index of the filter, and the y-axis represents the frequency. The model used for visualization is 3^9-DCNN with 59049 samples as input. Visualization was performed using the gradient ascent method to obtain the input waveform that maximizes the activation of a filter in the layers. To effectively find the filter characteristics, we set the input waveform estimate to 729 samples which is close to a typical frame size.

we add the bottom gradients to the input with gradient normalization. By repeating this process several times, we can obtain the waveform that maximizes the target filter activation. Examples of learned filters at each layer are in Figure 3. Although we can find the patterns that low-frequency filters are more visible along the layer, estimated filters are still noisy. To show the patterns more clearly, we visualized them as spectrum in the frequency domain and sorted them by the frequency of the peak magnitude.

Note that we set the input waveform estimate to 729 samples in length because, if we initialize and back-propagate to the whole input size of the networks, the estimated filters will have large dimensions such as 59049 samples in computing spectrum. Thus, we used the smaller input samples which can find the filter characteristics more effectively and also are close to a typical frame size in spectrum.

The layer 1 shows the three distinctive filter bands which are possible with the filter length with 3 samples (say, a DFT size of 3). The center frequency of the filter banks increases linearly in low frequency filter banks but it becomes non-linearly steeper in high frequency filter banks. This trend becomes stronger as the layer goes up. This nonlinearity was found in learned filters with a frame-level end-to-end learning [11] and also in perceptual pitch scales such as mel or bark.

6. CONCLUSION AND FUTURE WORK

In this paper, we proposed sample-level DCNN models that take raw waveforms as input. Through our experiments, we showed that deeper models (more than 10 layers) with a very small sample-level filter length and subsampling length are more effective in the music auto-tagging task and the results are comparable to previous state-of-the-art performances on the two datasets. Finally, we visualized hierarchically learned filters. As future work, we will analyze why the deep sample-level architecture works well without input normalization and nonlinear function that compresses the amplitude and also investigate the hierarchically learned filters more thoroughly.

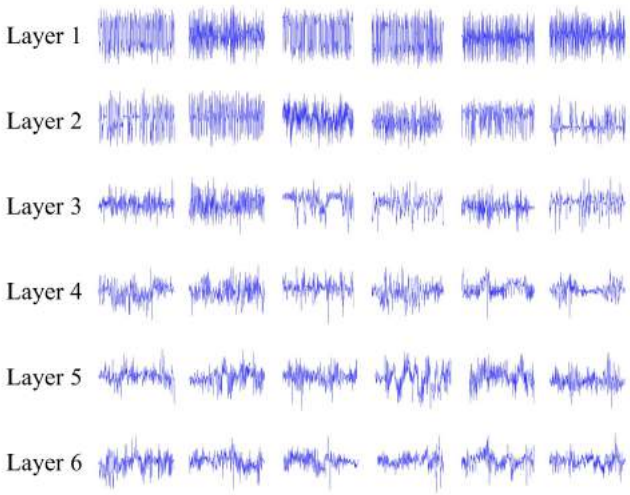

Figure 3. Examples of learned filters at each layer.

Acknowledgments

This work was supported by Korea Advanced Institute of Science and Technology (project no. G04140049) and National Research Foundation of Korea (project no. N01160463).

7. REFERENCES

[1] 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.

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

[3] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[4] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

[5] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," *arXiv preprint arXiv:1508.06615*, 2015.

[6] D. Palaz, M. M. Doss, and R. Collobert, "Convolutional neural networks-based continuous speech recognition using raw speech signal," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4295–4299.

[7] D. Palaz, R. Collobert *et al.*, "Analysis of cnn-based speech recognition system using raw speech as input," Idiap, Tech. Rep., 2015.

[8] R. Collobert, C. Puhrsch, and G. Synnaeve, "Wav2letter: an end-to-end convnet-based speech recognition system," *arXiv preprint arXiv:1609.03193*, 2016.

[9] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *CoRR abs/1609.03499*, 2016.

[10] T. N. Sainath, R. J. Weiss, A. W. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform cldnns." in *INTERSPEECH*, 2015, pp. 1–5.

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

[12] D. Ardila, C. Resnick, A. Roberts, and D. Eck, "Audio deepdream: Optimizing raw audio with convolutional networks."

[13] J. Pons, T. Lidy, and X. Serra, "Experimenting with musically motivated convolutional neural networks," in *IEEE International Workshop on Content-Based Multimedia Indexing (CBMI)*, 2016, pp. 1–6.

[14] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," in *Proceedings of the 17th International Conference on Music Information Retrieval (ISMIR)*, 2016, pp. 805–811.

[15] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," *arXiv preprint arXiv:1609.04243*, 2016.

[16] P. Hamel, S. Lemieux, Y. Bengio, and D. Eck, "Temporal pooling and multiscale learning for automatic annotation and ranking of music audio," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, 2011.

[17] J. Lee and J. Nam, "Multi-level and multi-scale feature aggregation using pre-trained convolutional neural networks for music auto-tagging," *arXiv preprint arXiv:1703.01793*, 2017.

[18] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of algorithms using games: The case of music tagging," in *ISMIR*, 2009, pp. 387–392.

[19] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, vol. 2, no. 9, 2011, pp. 591–596.

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

[21] J.-Y. Liu, S.-K. Jeng, and Y.-H. Yang, "Applying topological persistence in convolutional neural network for music audio signals," *arXiv preprint arXiv:1608.07373*, 2016.

[22] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, vol. 1341, p. 3, 2009.

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

[24] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 427–436.