DEEP LEARNING FOR MUSICAL INSTRUMENT RECOGNITION

Mingqing Yun
Department of Electrical and Computer Engineering
University of Rochester
myun5@ur.rochester.edu

Jing Bi
Department of Electrical and Computer Engineering
University of Rocheester
jbi5@ur.rochester.edu

ABSTRACT

The focus of this paper is to compare a convolutional neural network (CNN) and a recurrent neural network (RNN) in the particular task of instrument classification with log mel-spectrogram. We first choose to use a simple but efficient CNN architectureLeNet to verify the validity of using CNN for instrument classification. We propose a design strategy meant to capture the relevant time-frequency contexts for learning timbre, which permits using domain knowledge for designing architectures. In addition, another goal of this paper is to use one of RNN structure called Long-Short Term Memory to realize instrument recognition. After comparing different network structure, we can make a conclusion that the LeNet learns faster and more accurate when doing instrument classification.

1. INTRODUCTION

Our goal is to compare the performance of different deep learning architectures on recognizing instrument music signal. Different instrument has particular ”color” or the ”quality” of a sound.It has been found to be related to the spectral envelope shape and to the time variation of spectral content. [5]

Convolutional neural networks (CNNs) have been actively used for various music classification tasks such as music tagging [6] [3], genre classification [15] [2], and user-item latent feature prediction for recommendation [16]. most previous methodology require a dual pipeline:first,descriptors need to be extracted using a predefined algorithm and parameters; and second, temporal models require an additional tied on top of the proposed descriptor. Therefore, descriptors and temporal models are typically not jointly designed. Throughout this study, we explore recognizing instrument by deep learning with the input set to be log magnitude spectrogram. CNNs features in different levels of hierarchy and can be extracted by convolutional kernels. The hierarchical features are learned to achieve a given task during supervised training. For example, learned features from a CNN that is trained for genre classification low-level features (e.g., onset) to high-level features (e.g., percussive instrument patterns) [4]. This end-to-end learning approach allows minimizing the effect of fixed steps. Note that no strong assumptions over the descriptors are required since a generic perceptually-based pre-processing is used: log magnitude spectrograms.

Identifying sound is an inherently temporal task, and some previous work indicates that classification of instrument may depend on temporal feature. An effective way to model temporal processing is by using recurrent neural networks (RNNs), which learn representations from sequential data [7]. As its name indicates, a RNN processes the incoming input by also considering its own output given the history input. In AI, RNNs have made impressive progress in speech and action recognition [1], demonstrating the potential to use temporal feature for classification. Meanwhile, RNN can be interpreted as a temporal model (if more than one frame is input to the network) that allows learning spectro-temporal descriptors from spectrograms. In this case, learnt descriptors and temporal model are jointly optimized, what might imply an advantage when compared to previous methods. Therefore, in this paper we test a suitable classifier, called Long Short Term Memory (LSTM), which is a Recurrent Neural Network (RNN) that allows to deal with actual temporal patterns. In order to compare with CNN, same spectrograms were used to train and test RNN.

From the different deep learning approaches, we focus on CNNs and RNNs due to several reasons:

1. by taking spectrograms as input, one can interpret filter dimensions in time-frequency domain;
2. and they both can efficiently exploit invariance such as time and frequency invariance present in spectrograms by sharing a reduced amount of parameters.
3. RNNs are flexible in selecting how to summarize the local features, which can be helpful with extracting temporal feature.
4. CNNs have better performance with local feature extraction.

Additionally, most CNN architectures use unique filter shapes in every layer [6] [8]. recent works point out that using different filter shapes in each layer is an efficient way to exploit CNN’s capacity [13] [14]. For example, Pons et al. [14] proposed using different musically motivated filter shapes in the first layer to efficiently model several musically relevant time-scales for learning temporal features.

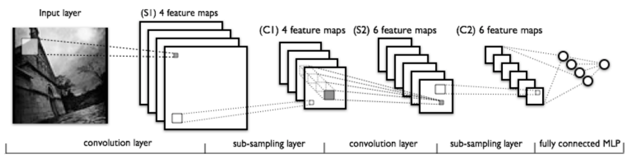


Figure 1. LeNet model [12]

This paper is organized as follows. Section 2 and Section 3 briefly described CNN and RNN architecture and different aspects they extract feature form dataset Section 4 focus on the data preprocessing, experiment and result followed with conclusion in 5.

2. NETWORK ARCHITECTURE

2.1 CNN architecture

First we use LeNet to test the feasibility of using CNN to classify instrument. The architecture we choose to use is LeNet [12], which is a relative simple but efficient network. Figure 1 shows a graphical depiction of a LeNet model.

As the figure shows, sparse, convolutional layers and max-pooling are at the heart of the LeNet models. The lower-layers are composed to alternating convolution and max-pooling layers. The upper-layers however are fully-connected and correspond to a traditional MLP (hidden layer + logistic regression). The input to the first fully-connected layer is the set of all features maps at the layer below. This kind of architecture is very useful for extract local feature and classifying the data. Besides, the ”color” of the instrument is found to be related to the spectral envelope shape and to the time variation of spectral content.

Therefore, it is reasonable to assume different instrument has its own time-frequency expression, which means the shape of the kernel is important for classification task. Equation below indicates the way next layer get information from previous layer.

$$x_l^j = f(\sum_{i \in M_j} x_i^{l-1} * k_{ij}^l + b_j^l) \qquad (1)$$

Form this equation, we can observe that the shape of kernel is crucial for information extraction. In order to focus on how to exploit the capacity of spectrograms to represent instrument, we choose two different convolutional kernel shapes(5*5,3*8)using same architecture.

In Figure 2, the design strategy allows to efficiently model different musically relevant time-frequency contexts. Moreover, this design strategy ties very well with the idea of using the available domain knowledge for designing filter shapes that can intuitively guide the different filter shapes design so that spectro-temporal envelopes can be efficiently represented within a single filter.

2.2 RNN architecture

As mentioned above, instrument is not classified only rely on their spectral pattern,but also on temporal patterns.

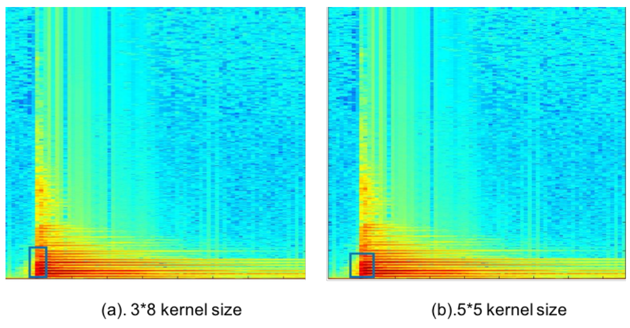


Figure 2. kernel shape model as 3×8 and 5×5

therefore, it is also important to discuss the influence of temporal feature. we build up a network with one simple layer of LSTM as shown in Figure 3

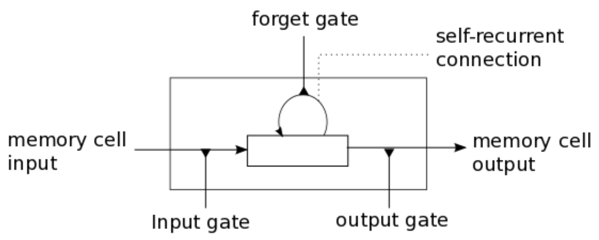


Figure 3. Example of a figure caption [11]

In this paper we choose a new and promising model of recurrent neural network called Long Short Term Memory (LSTM) [9]. LSTM is an RNN that uses self-connected unbounded internal memory cells protected by nonlinear multiplicative gates.

Error is back-propagated through the network in such a way that exponential decay is avoided. The unbounded (i.e. unsquashed) cells are used by the network to store information over long time durations. The gates are used to aid in controlling the flow of information through the internal states. The cells are organized into memory blocks, each having an input gate that allows a block to selectively ignore incoming activations, an output gate that allows a block to selectively take itself off-line, shielding it from error, and a forget gate that allows cells to selectively empty their memory contents. The cell blocks are basically a replacement of original RNN's hidden layer. The forward output of cells, combine value of output gate and activated cell value,shows below:

$$b_c^t = b_\omega^t h(s_c^t) \qquad (2)$$

Where b_ω^t is from the output gate, s_c^t is a state value from the cell. Note that each memory block can contain several memory cells. Each gate has its own activation in the range [0,1]. Moreover, the backward output of cell, which is

$$\epsilon_c^t = \frac{\partial L}{\partial b_c^t} \qquad (3)$$

$$\epsilon_c^t = \sum_{k=1}^{K} \omega_{ck} \delta_t^k + \sum_{g=1}^{G} \omega_{cg} \delta_g^{t+1} \tag{4}$$

By using gradient descent to optimize weighted connections into gates as well as cells, an LSTM network can learn to control information flow. Furthermore, we also build a two-layer LSTM to compare with one-layer LSTM. The main goal here is to observer whether the performance of classification task can be improved by simply adding one additional layer.

3. EXPERIMENT

This section describes the proposed approach in instrument classification. In order to test the proposed method, we run a series of experiments on a chosen dataset. We evaluate the results by comparing the system outputs to the annotated references.

3.1 Dataset

To evaluate our system, we collect data samples from the internet to make a instrument dataset. The instrument dataset [10]used is annotated with the name of the instrument. We choose 14 instruments out of 25 with 200 training and 120 test audio . Each audio is trimmed into 1 second, and the start point of the clip is choose from the maximum of the derivative of the signal power.

3.2 Processes

We use log mel power as acoustic features.We first cut all the recordings into 1 second. Then we compute short-time Fourier transform(STFT) of the recordings with 1024 fft length and 50% overlap. We set filterbank with 64 and 128 bands spanning 0 to 22050Hz, which is the Nyquist rate. Finally we computing dB relative to peak power and nominalized the data.

By setting different number of filterbank , we can determine whether the number of filterbank can improve the accuracy. The result shows in section 4.4.1.

3.3 Training network

We train two different networks and evaluate their results. Since the size of the spectrogram can affect the network architecture, we use spectrogram prosessed by 64 filterbank in 4.2, and only first 80 frames preserved.

3.3.1 CNN

We choose Lenet to evaluate CNN functions. The architecture shows in Figure 4. The input size of the spectrogram is 64×80. The kernel size of the first CNN layer is 5×5 with stride to be 1. Next, it passes through a maxpooling layer with 2×2 kernel size and the stride is 2. After that, another CNN layer as well as another maxpooling layer has been set. Finally, 3 fully connected layer has been set, the units among them are 120, 84, 14.

In order to determine whether different size of kernel may affect the final result. We also set a contrast test which change the kernel size in CNN layer in to 3×8. The result in section 4.4.2.

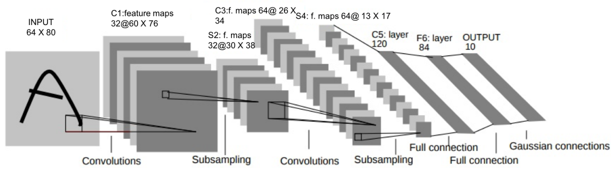


Figure 4. Lenet architecture with parameters

3.3.2 LSTM

For LSTM, we set 3 different comparative test in order to get the best results.

Firstly, with only 1 LSTM layer, we set 64 units compares to 128 units, where the units means the output dimension. After connected to a fully connected layer which has 14 units, we can determined whether the number of units affects the result. The result shows in section 4.4.3.

Secondly, in order to determinate whether the number of layer affects the result, we set a 1 LSTM layer system compares to a 2 LSTM layer system. The 1 LSTM layer system contains 64 units and the 2 LSTM layer system contains 128 units in each layer. The result shows in section 4.4.4

Thirdly, with 2 LSTM layers, we set two different systems. The first system contains 128 units in the first layer and 128 units in the second layer. The second system contains 128 units in the first layer and 64 units in the second layer. By comparing the result from two systems, we can determine whether the number of units affects the result. The result in section 4.4.5

3.4 Result

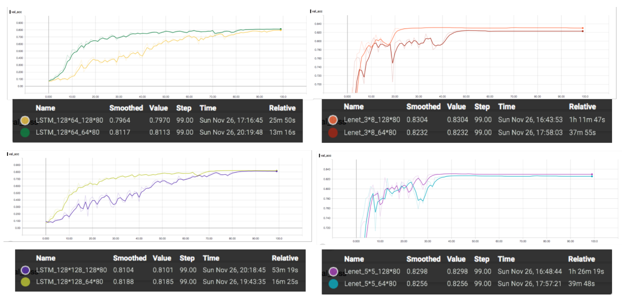
To evaluate the result, we use traceback function in tensorflow to help observe the leaning rate and test accuracy. All result comes out of 100 epoch. The x_axis of figure 5 - 10 is the number of epoch, and the y_axis of figure 5 - 10 is accuracy.

3.4.1 Different number of filterbank comparation

We train two different experiments for each network.The result in Figure 5. The labels are present as "network name_ units in each layer(or filter size)_ input size(128*80 or 64*80)".The result shows that in an LSTM system, whether the units in different layers are the same or not, the 64*80 size input always get a better result. But for LeNet system, whether the filter size is 3*8 or 5*5, the 128*80 size input always get a better result.

3.4.2 Filter size comparation in LeNet system

With the input size 64*80, we set a contrast test which change the kernel size in CNN layer from 5*5 into 3*8. As shown in Figure 6, the results come out of the 100 epoch are almost the same. But when in the first epoch, we notice that with a 3*8 kernel, the LeNet learns faster than with a 5*5 kernel.



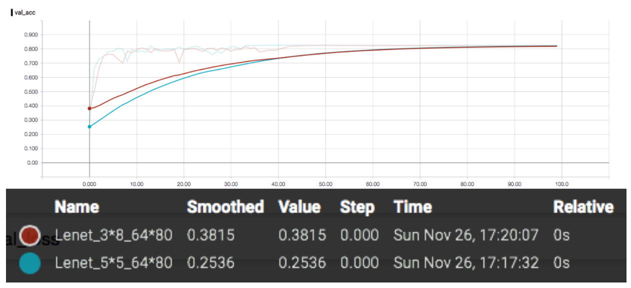
	Name	Smoothed	Value	Step	Time	Relative
●	LSTM_128*64_128*80	0.7964	0.7970	99.00	Sun Nov 26, 17:16:45	25m 50s
●	LSTM_128*64_64*80	0.8117	0.8113	99.00	Sun Nov 26, 20:19:48	13m 16s

	Name	Smoothed	Value	Step	Time	Relative
●	Lenet_3*8_128*80	0.8304	0.8304	99.00	Sun Nov 26, 16:43:53	1h 11m 47s
●	Lenet_3*8_64*80	0.8232	0.8232	99.00	Sun Nov 26, 17:58:03	37m 55s

	Name	Smoothed	Value	Step	Time	Relative
●	LSTM_128*128_128*80	0.8104	0.8101	99.00	Sun Nov 26, 20:18:45	53m 19s
●	LSTM_128*128_64*80	0.8188	0.8185	99.00	Sun Nov 26, 19:43:35	16m 25s

	Name	Smoothed	Value	Step	Time	Relative
●	Lenet_5*5_128*80	0.8298	0.8298	99.00	Sun Nov 26, 16:48:44	1h 26m 19s
●	Lenet_5*5_64*80	0.8256	0.8256	99.00	Sun Nov 26, 17:57:21	39m 48s

Figure 5. Result from different preprocess method



	Name	Smoothed	Value	Step	Time	Relative
●	Lenet_3*8_64*80	0.3815	0.3815	0.000	Sun Nov 26, 17:20:07	0s
●	Lenet_5*5_64*80	0.2536	0.2536	0.000	Sun Nov 26, 17:17:32	0s

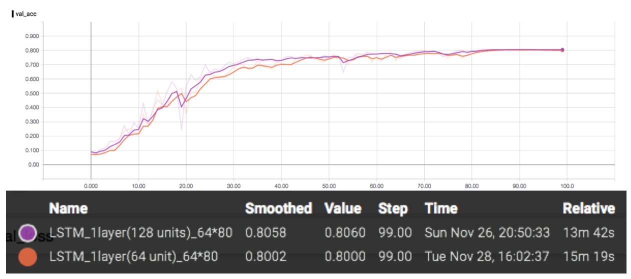
Figure 6. Result from LeNet system with 3*8 and 5*5 kernel size

3.4.3 Number of units comparation in LSTM system

With the input size 64*80, we set a system with 128 units in the LSTM layer compares to a system with 64 units in the LSTM layer. As shown in Figure 7, we can make a conclusion that larger number of units runs more effectively, not only test accuracy but also running time.

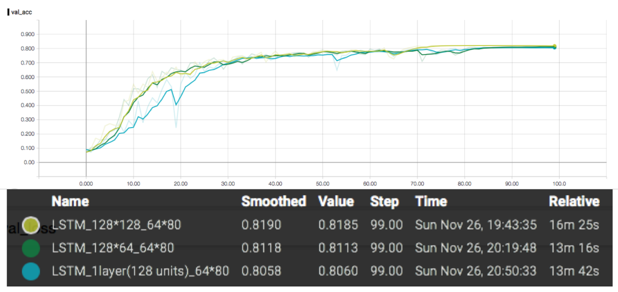
3.4.4 Number of layers comparation in LSTM system

In this part, we set three different experiments. One of the experiment is a one LSTM layer system with 128 units. The other two systems are two layers system. One contains 128 units in the first layer and 128 in the second layer. The other one contains 128 layers in the first layer and 64 in the second layer. The result shows in Figure 8. We can clearly observe that with input size 64*80, the two layers systems always get a better result compares to the one layer system.



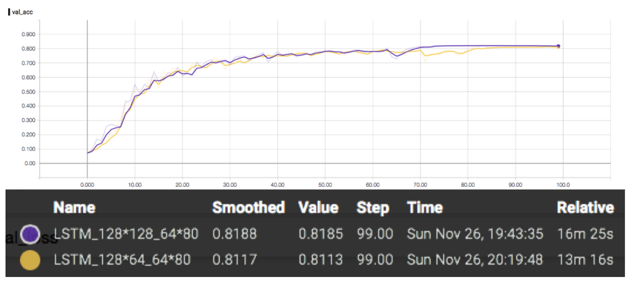
	Name	Smoothed	Value	Step	Time	Relative
●	LSTM_1layer(128 units)_64*80	0.8058	0.8060	99.00	Sun Nov 26, 20:50:33	13m 42s
●	LSTM_1layer(64 unit)_64*80	0.8002	0.8000	99.00	Tue Nov 28, 16:02:37	15m 19s

Figure 7. Result from a one LSTM layer system with 128 and 64 units



	Name	Smoothed	Value	Step	Time	Relative
●	LSTM_128*128_64*80	0.8190	0.8185	99.00	Sun Nov 26, 19:43:35	16m 25s
●	LSTM_128*64_64*80	0.8118	0.8113	99.00	Sun Nov 26, 20:19:48	13m 16s
●	LSTM_1layer(128 units)_64*80	0.8058	0.8060	99.00	Sun Nov 26, 20:50:33	13m 42s

Figure 8. Result from 1 layer system and 2 layers systems



	Name	Smoothed	Value	Step	Time	Relative
●	LSTM_128*128_64*80	0.8188	0.8185	99.00	Sun Nov 26, 19:43:35	16m 25s
●	LSTM_128*64_64*80	0.8117	0.8113	99.00	Sun Nov 26, 20:19:48	13m 16s

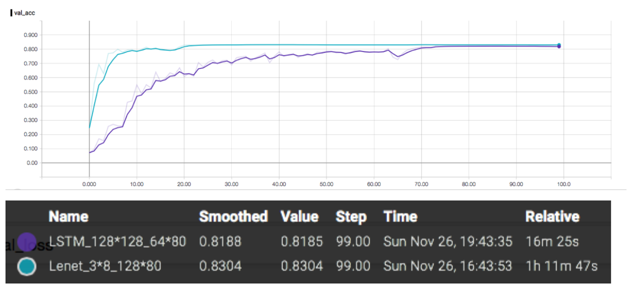
Figure 9. Result from 128 units and 64 units in the second LSTM layer

3.4.5 Number of units in the second LSTM layer comparation

In this part, we set two comparative systems. One of them contains 128 units in the second LSTM layer. The other contains 64 units in the second LSTM layer. All the other settings are the same. As shown in Figure 9, we observe that the result shows limit difference.

3.4.6 LeNet compares with LSTM

After all the test above, we can successfully choose the best result from either the CNN system and LSTM system. As for a CNN system, the LeNet structure with 3*8 kernel size and 64*80 input size gets the best result. As for a LSTM system, the two layer system with 128*80 input size and each layer contains 128 units gets the best result. The result shows in Figure 10.



	Name	Smoothed	Value	Step	Time	Relative
●	LSTM_128*128_64*80	0.8188	0.8185	99.00	Sun Nov 26, 19:43:35	16m 25s
●	Lenet_3*8_128*80	0.8304	0.8304	99.00	Sun Nov 26, 16:43:53	1h 11m 47s

Figure 10. Result from LeNet system compares with LSTM system

4. CONCLUSION

From the results shows in 3.4, we notice that in a LeNet system, the 3*8 kernel size can get a better solution. And in a LSTM system, not only the number of units, but also the number of layers can affect the system. After several comparation, we notice that a two layer LSTM system with 128 units in each layer gets the best solution. As for the size of input data, we notice that the LeNet is more suitable with larger input size(128*80).But the LSTM get better result with 64*80 input size. This may because LeNet is a well developed neural network system which can deal with more information.

In the last experiment , we compare LeNet system to LSTM system. As shown in Figure 10, we can clearly see that the learning rate from LeNet are faster than from LSTM system. And after 100 epochs, the result from LeNet is a little bit higher than from LSTM system. In this case, we can make a conclusion that LeNet system is more effective when classifying music instrument.

5. ACKNOWLEDGEMENT

We would like to give special thanks to Dr.Yichi Zhang and Prof.Duan for their guidance. Also, thank three reviewers for their suggestions in improving this article.

6. REFERENCES

[1] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating Sentences from a Continuous Space. *arXiv:1511.06349 [cs]*, November 2015. arXiv: 1511.06349.

[2] P. Chiliguano and G. Fazekas. Hybrid music recommender using content-based and social information. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2618–2622, March 2016.

[3] Keunwoo Choi, George Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. *arXiv:1606.00298 [cs]*, June 2016. arXiv: 1606.00298.

[4] Keunwoo Choi, George Fazekas, and Mark Sandler. Explaining Deep Convolutional Neural Networks on Music Classification. *arXiv:1607.02444 [cs]*, July 2016. arXiv: 1607.02444.

[5] Keunwoo Choi, George Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional Recurrent Neural Networks for Music Classification. *arXiv:1609.04243 [cs]*, September 2016. arXiv: 1609.04243.

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

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, November 2016. Google-Books-ID: omivDQAAQBAJ.

[8] Yoonchang Han, Jaehun Kim, Kyogu Lee, Yoonchang Han, Jaehun Kim, and Kyogu Lee. Deep Convolutional Neural Networks for Predominant Instrument Recognition in Polyphonic Music. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 25(1):208–221, January 2017.

[9] Sepp Hochreiter and Jrgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[10] Peter Isley. Sound Samples Philharmonia Orchestra, 2008. [Online; accessed 19-July-2008].

[11] J. Kim, J. Kim, H. L. T. Thu, and H. Kim. Long short term memory recurrent neural network classifier for intrusion detection. In *2016 International Conference on Platform Technology and Service (PlatCon)*, pages 1–5, Feb 2016.

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[13] Huy Phan, Lars Hertel, Marco Maass, and Alfred Mertins. Robust Audio Event Recognition with 1-Max Pooling Convolutional Neural Networks. *arXiv:1604.06338 [cs]*, April 2016. arXiv: 1604.06338.

[14] Jordi Pons and Xavier Serra. Designing efficient architectures for modeling temporal features with convolutional neural networks. In *42th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2017). IEEE, New Orleans, USA*, 2017.

[15] S. Sigtia and S. Dixon. Improved music feature learning with deep neural networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6959–6963, May 2014.

[16] Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2643–2651. Curran Associates, Inc., 2013.