Interspeech 2018
2-6 September 2018, Hyderabad

Impact of Aliasing on Deep CNN-Based End-to-End Acoustic Models

Yuan Gong, Christian Poellabauer

Department of Computer Science and Engineering
University of Notre Dame, IN 46556, USA
ygong1@nd.edu, cpoellab@nd.edu

Abstract

A recent trend in audio and speech processing is to learn target labels directly from raw waveforms rather than hand-crafted acoustic features. Previous work has shown that deep convolutional neural networks (CNNs) as front-end can learn effective representations from the raw waveform. However, due to the large dimension of raw audio waveforms, pooling layers are usually used aggressively between temporal convolutional layers. In essence, these pooling layers perform operations that are similar to signal downsampling, which may lead to temporal aliasing according to the Nyquist-Shannon sampling theorem. This paper explores, using a series of experiments, if and how this aliasing effect impacts modern deep CNN-based models.
Index Terms: aliasing, CNNs, end-to-end learning

1. Introduction

Conventional audio and speech analysis systems are typically built using a pipeline structure, where the first step is to extract various low dimensional hand-crafted acoustic features, e.g., Mel frequency scale cepstral coefficients (MFCC) or features extracted using linear predictive coding (LPC) and perceptual linear prediction (PLP) [1]. A problem with hand-crafted features is that it is not possible to retain all useful information with a limited set of features and that these features are not always best for the classification objective at hand. To overcome these limitations, several prior efforts began to abandon handcrafted features and instead feed raw magnitude spectrogram features directly into the deep convolutional neural networks (CNNs) or deep recurrent neural networks (RNNs) [2, 3, 4, 5]. Furthermore, another recent trend in audio and speech processing is the learning directly from raw waveforms, which provides a more thorough end-to-end process by completely abandoning the feature extraction step. These networks typically consist of one or more temporal convolutional blocks (i.e., a 1-D temporal convolutional layer followed by a nonlinear activation function and a pooling layer) as the front-end, followed by decision-making layers, e.g., RNNs or fully connected layers. The parameters of these convolutional layers are then learned jointly with the rest of the network using optimization algorithms. Studies have shown that such 1-D temporal convolutional blocks are capable of approximating standard filterbanks, such as a gammatone filterbank [6, 7, 8]. For example, in [9, 10], the authors use one temporal convolutional layer and then conduct a global averaging pooling throughout each window of 25-35ms, and then feed the outputs into RNNs. Other projects stack multiple time-domain convolutional blocks in sequence [6, 11, 12, 13] to learn the representation of the raw waveform. In [14, 15], the authors explore the use of very deep network architectures with 8 and 34 temporal convolutional blocks, respectively.

Using multiple convolutional blocks to learn a representation seems natural, because such architectures have been used successfully in computer vision tasks [16, 17]. For computational efficiency, deep CNNs usually use fewer filters in earlier convolutional layers [14, 15]. From the perspective of signal processing, this can be viewed as a multi-level filtering operation: the first temporal convolutional layer performs primary filtering of the input waveform; the output signal is then fed into a pooling layer followed by another convolutional layer, which then performs additional filtering. However, performing temporal pooling (i.e., pooling over time) is similar to a downsampling operation in signal processing. According to the Nyquist–Shannon sampling theorem, downsampling will lead to the **aliasing effect**, where the frequency components that are more than half of the new sampling rate will be mistakenly sampled as low-frequency components and mixed into the real low-frequency components. Therefore, the convolutional layers after the pooling layer actually receive an aliased signal from the preceding network where aliased high-frequency components and the actual low-frequency components were mixed together and have become indistinguishable. This raises the questions of whether a succeeding convolutional layer will be able to perform effective filtering, whether this aliasing effect will impact the performance of deep learning models, and if so, what the extent of this impact will be. In this work, we provide a focused exploration and discussion of these questions, which, to the best of our knowledge, have not been extensively studied before.

2. The Impact of Aliasing on Deep CNNs

In order to facilitate analysis and discussion, we use a simple network architecture with two temporal convolutional blocks for our experiments. As shown in Figure 1, first, we take a small window of the raw waveform (consisting of M samples) and convolve the raw waveform with a set of N_1 filters. We use the same padding to ensure that the output from the convolution is $M \times N_1$. We then apply an activation function to the output signal and perform a pooling of size K. The output from the pooling layer will be $\lceil \frac{M}{K} \rceil \times N_1$. Finally, we pool the output in time over the entire window, to produce $1 \times N_2$ outputs. We then shift the window and repeat this process. For an input signal with $T \times M$ points, the output will be a $T \times N_2$ "time-frequency" representation. We then pass it to the succeeding layers. We refer to layers before this time-frequency representation as "front-end layers" and layers afterwards as "back-end layers". The back-end layers include multiple 2-D time-frequency convolutional blocks and two fully connected layers. Each 2-D convolutional block consists of a convolutional layer with 32 filters, each of size [2,2], and a max pooling layer with pooling size [2,2]. We stack three such blocks in experiments 1, 2, and 4; and 2 blocks in experiment 3, according to the input length. The dense layers have 128 and 64 units, respectively. The design of the back-end architecture is similar to the one presented in [3]. In our experiments, this back-end ar-

10.21437/Interspeech.2018-1371

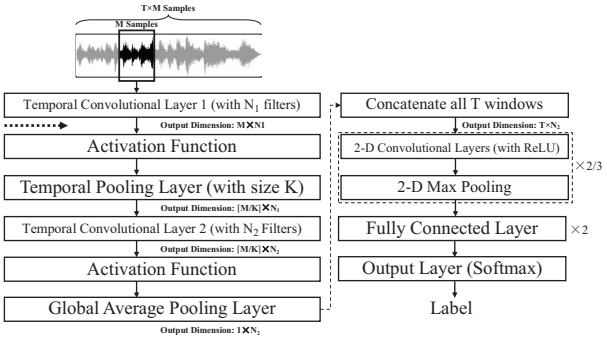

Figure 1: *The network architecture used in our experiments.*

chitecture performs slightly better than stacking LSTM layers.

Since our focus is on the impact of the aliasing effect caused by the temporal pooling layer between adjacent temporal convolutional layers, we want to primarily investigate how the aliasing effect depends on the pooling type, the size K of the pooling layer between the temporal convolutional layers, and the number of filters in the convolutional layer before the pooling layer. Since global average pooling can be considered as an estimate of the energy of the output temporal signal throughout the entire window, which discards temporal details, we do not study the aliasing effect in the back-end layers. The following parameters are kept identical across all experiments: the input signal is sampled at 16kHz, the window size $M = 640$ (which corresponds to 40ms), the convolutional filter size is 256, and the number of filters of the second convolutional layer $N_2 = 16$. Further, the learning rate is selected using a grid search in [1e-5, 5e-5, 1e-4, 5e-4, 1e-3, 5e-3] and the network is optimized using an Adam optimizer [18] with a Xavier uniform initializer [19]. The training set and test set are independent and the model is selected according to the performance on a small independent validation set, which is reserved from the training set.

2.1. The Aliasing Effect

According to the Nyquist-Shannon sampling theorem, signal downsampling leads to the aliasing effect, i.e., the frequency f that is over half of the new sampling rate f_s will be mistakenly sampled as low aliased frequency f_a:

$$f_a = \left| f - \frac{(k+1)f_s}{2} \right| \quad \text{where} \quad \frac{kf_s}{2} \leq f \leq \frac{(k+2)f_s}{2}$$

However, in modern neural networks, direct downsampling is rarely used. Instead, the output temporal signal of preceding convolutional layers is first applied with an activation function and then compressed using max pooling or average pooling. The practical compression operation that combines the activation function and max/average pooling is not the same as a single downsampling operation. Therefore, the first question we explore is: **will such practical compression operations also lead to the aliasing effect?** That is, are signals with mirrored frequencies distinguishable after applying the activation function and max/average pooling? In this section, we discuss a typical case where the original sampling rate is 16kHz and the pooling size is 2, i.e., the *folding frequency* is 4kHz. We further refer to the frequencies that are symmetrical around the folding frequency as *mirrored frequencies* and the corresponding signals as *mirrored signal pairs*. We first consider the setting currently most deep neural network (DNN) models use: the ReLU activation function [20] and max pooling. From the perspective of signal processing, ReLU is equivalent to a half-wave rectifier,

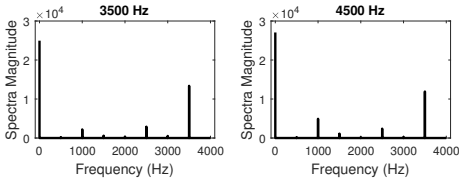

Figure 2: *Comparison of the spectrograms of two mirrored signals of same magnitude after ReLU and max pooling.*

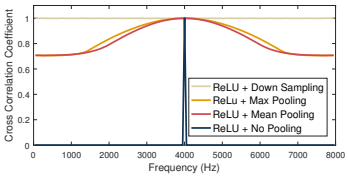

Figure 3: *The CCCs of signals with different frequencies and their mirrored signals after ReLU and pooling.*

which will add harmonic frequency components to the output signal. Further, we empirically found that after max pooling, the mirrored signal pair has main components and harmonic components at the same frequency points, but the magnitudes are different (shown in Figure 2). If we measure the similarity of mirrored signal pairs using the cross-correlation coefficient (CCC) (shown in Figure 3), we can observe the phenomenon that the closer the mirrored signal pairs are to the folding frequency, the higher the similarity they have. The magnitude ratio and phase difference of harmonic frequency components and the main frequency component shows a pattern that can be used to distinguish mirrored signal pairs. However, the magnitude of harmonic frequency components is small and practical audio signals do not consist of only a single frequency component, therefore, the harmonic frequency components are easy to be mixed and submerged in the background broad spectrum. Thus, the mirrored signal pairs are hard to distinguish after a compression of ReLU and max pooling when they are close to the folding frequency. A theoretical analysis of other non-linear activation functions and pooling methods is difficult. Therefore, we execute the following experiment for validation:

Experiment 1: Given a label set of frequency points $\boldsymbol{C} = \{f_1, f_2, ..., f_n\}$, we synthesize a set of 1-second signals $\boldsymbol{S}$, each signal composed of one main frequency component $f \in \boldsymbol{C}$ with a magnitude randomly selected from [0.1,1.0] and 128 noise frequency components at random frequency points with magnitudes randomly selected from [0, 0.1] in the frequency domain. The DNNs are asked to recognize the frequency of the main components, i.e., to find a mapping $\boldsymbol{S} \mapsto \boldsymbol{C}$. For each f, we synthesize 1000 signal samples for training and 32 signal samples for testing.

In this experiment, we want to see that if the mirrored signal pairs output from the preceding convolutional layer can be distinguished from the rest of the network after the activation function and pooling layer. Hence, we input the samples right after the first convolutional layer and before the activation function (the dashed arrow in Figure 1). The sample rate of the synthetic signals is 16kHz. We then perform experiments with a set $\boldsymbol{S_1}$ (corresponding to $\boldsymbol{C_1}$, which consists of frequencies ranging from 3.5kHz to 4.5kHz with an interval of 0.1kHz except for the folding frequency 4kHz (10 classes)), and a set $\boldsymbol{S_2}$ (corresponding to $\boldsymbol{C_2}$, which consists of frequencies ranging

Table 1: *The classification accuracy (%) for $\boldsymbol{S_1}$ (first number) and $\boldsymbol{S_2}$ (shown in parentheses) in experiment 1.*

	ReLU	Linear	Sigmoid
Downsampling	45.4 (91.0)	54.5 (93.4)	55.3 (97.7)
Average Pooling	49.4 (91.4)	53.1 (94.8)	58.2 (93.7)
Max Pooling	47.2 (93.8)	54.8 (93.7)	69.0 (92.6)

Figure 4: *The confusion matrix of experiment 1 for $\boldsymbol{S_1}$ (left) and $\boldsymbol{S_2}$ (right) when using ReLU and max pooling.*

from 3.5Hz to 4.0kHz with an interval of 0.1kHz and 4.05kHz to 4.45kHz with an interval of 0.1kHz (10 classes)). $\boldsymbol{S_1}$ includes five pairs of mirrored signals after 2-pooling, while $\boldsymbol{S_2}$ includes no mirrored signals. We perform the tests with different activation functions and pooling layers in the first convolutional block (ReLU and max pooling are used in other layers).

The results are shown in Table 1. The test for $\boldsymbol{S_2}$ shows a classification accuracy of over 91% (which means that the DNNs are capable of this task), while the test for $\boldsymbol{S_1}$ achieves only around 50%. Since the result is similar for all activation and pooling layer settings, except that the Sigmoid function and max pooling displays better accuracy, and considering that the Sigmoid function is seldomly used in modern DNNs due to the vanishing gradient problem, we use the commonly-used ReLU and max pooling setting as representative for our discussions in the rest of this paper. The confusion matrix (shown in Figure 4) shows that the most classification errors happen in mirrored signal pairs and very few errors are across different mirror pairs. This shows that the DNNs cannot distinguish mirrored signal pairs, but have no problem in classifying non-mirrored signals in $\boldsymbol{S_1}$. Nevertheless, the above experiments were performed near the folding frequency, which according to Figure 3 has a high similarity. Next, we demonstrate that mirrored signal pairs that are farther from the folding frequency are also distinguishable by the DNNs. The next experiment explores the relationship of the DNNs' classification accuracy with regard to the CCC of the mirrored signals. The results are shown in Figure 5, where we can see that with decreasing CCC, the classification accuracy improves. More specifically for this example, in the range of 3.3kHz to 4.7kHz, the classification accuracy of DNNs is less than 70% (CCC≈0.97), while outside of this range, the mirrored signals begin to be distinguishable by the DNNs (we call this the *aliasing region*). Using CCC=0.97 as a threshold for aliasing, we calculate the aliasing region for other pooling sizes (shown in Figure 6). The aliasing regions do not overlap, even when the pooling size $K = 8$, which means that there exists at most one mirrored frequency for a given frequency, while simple downsampling will lead to $K - 1$ mirrored frequencies. In summary, the operation that combines the activation function and pooling layer will also lead to the aliasing effect, but will only impact a smaller frequency range near the folding frequency, compared to the single downsampling operation.

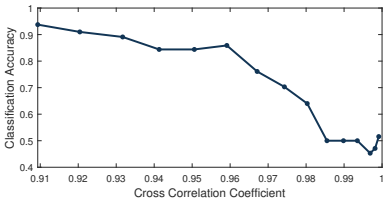

Figure 5: *Classification accuracy with regard to the CCC of mirrored signal pairs when using ReLU and max pooling.*

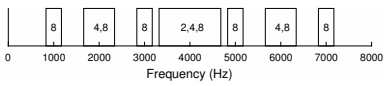

Figure 6: *The aliasing regions (rectangles) expand with increasing pooling size K (the numbers in the rectangles).*

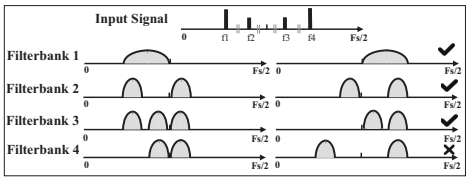

Figure 7: *Examples of the anti-aliasing filterbanks (marked with $\surd$). Note that although f_2 and f_3 are aliased in one filter of filterbank 3, they are seperated out in another filter, therefore the following operation will be able to distinguish them through further filtering and subtraction.*

2.2. The Anti-Aliasing Filterbank

Since the mirrored signal components in the output signal of the preceding convolutional layer cannot be distinguished by the succeeding layers after pooling, in order to avoid aliasing, the preceding layer needs to conduct effective anti-aliasing filtering, i.e., making the mirrored frequency components of the signal separable and then feed them to the succeeding layer. When the pooling size is two, theoretically, the preceding convolutional layer only needs two filters to avoid aliasing (e.g., a low-pass filter that stops at the folding frequency and a high-pass filter that starts at the folding frequency). We illustrate this in Figure 7. However, **do DNNs really learn such an anti-aliasing filterbank?** We perform an experiment to explore this question.

Experiment 2: We synthesize a set of signals in a similar manner as in experiment 1, except that we also add frequency components at all frequency points in the label set $\boldsymbol{C}$, while the frequency component at the main (label) frequency point has the largest magnitude. The model is asked to recognize the frequency of the main frequency component. We synthesize the sets $\boldsymbol{S_1'}$ and $\boldsymbol{S_2'}$ corresponding to the same label sets $\boldsymbol{C_1}$ and $\boldsymbol{C_2}$ as in experiment 1. But different from experiment 1, the signal is input to the front of the network. To accomplish this task, the model essentially needs to filter out all $f \in \boldsymbol{C}$ before the global average pooling layer and find the components with the largest magnitude. Since each signal sample in $\boldsymbol{S_1'}$ contains components at the mirrored frequency point of the main component, the DNNs further need to learn anti-aliasing filterbanks in the first convolutional layer. We repeat the experiment with different numbers of filters in the first convolutional layer N_1.

As shown in Table 2, we find that the DNNs perform well on $\boldsymbol{S_2'}$, even when $N_1 = 1$. In contrast, for $\boldsymbol{S_1'}$, which includes

Table 2: *The classification accuracy (%) for $\boldsymbol{S_1'}$ (first number) and $\boldsymbol{S_2'}$ (shown in parentheses) in experiment 2.*

# Filters of the First Convolutional Layer N_1				No Pooling
1	2	4	8	
47.9 (91.2)	59.4 (91.1)	63.0 (93.3)	77.1 (88.5)	93.8 (92.9)

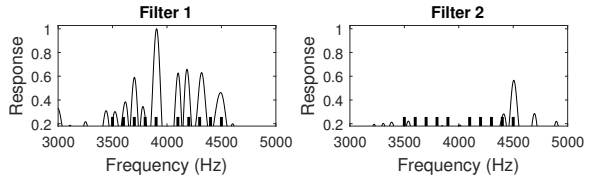

Figure 8: *Learned filter in the $\boldsymbol{S_1'}$ test when $N_1 = 2$. Bold lines represent frequency points $\in \boldsymbol{C}$.*

mirrored signal components, even when the first convolutional layer has more than two filters, the classification accuracy is still substantially worse than that for $\boldsymbol{S_2'}$. The frequency response of the filters of the first layer shows that they do not separate out the mirrored signal. For example, as shown in Figure 8, when $N_1 = 2$, filter 1 responds to all frequencies in $\boldsymbol{C_1}$, except for 4.4kHz, while filter 2 only responds to 4.4kHz and 4.5kHz. In this case, the signal components between 3.7kHz and 4.3kHz are aliased by filter 1 and not passed by filter 2, and are therefore difficult to be distinguishable in the successive filtering and operations. With an increasing number of filters in the first convolutional layer, the performance improves, but is still unsatisfactory. We perform another experiment for comparison, i.e., we eliminate the first pooling layer (therefore no aliasing) and obtain a classification accuracy of 93.8% on $\boldsymbol{S_1'}$ when $N_1 = 1$, which further proves that it is the aliasing that leads to the performance drop. In summary, although in theory the preceding convolutional layer can use anti-aliasing filters to eliminate the impact of aliasing, in practice it does not do so.

2.3. Impacts on Practical Tasks

In the previous section, we explored the impact of aliasing on a designed task, i.e., to recognize a simple frequency pattern from the waveform. In this task, the magnitude of the main frequency components is the indispensable clue for classification. We have seen that aliasing can greatly hurt the performance of DNNs by blurring this indispensable clue. **Will the aliasing effect also impact the performance of practical tasks?** Based on the earlier discussion, we believe that it depends on the task, because practical acoustic recognition tasks usually require recognition of a more complex time-frequency pattern, but this also means that more clues are given for inference. Assume that the target pattern is composed of several discriminative frequency components, then even if some are affected by aliasing, a prediction can still be made based on others. Further, the discriminative frequency components might not fall into the aliasing region. We perform the following experiment to investigate the impact on practical audio recognition tasks:

Experiments 3 & 4: We perform a phone recognition test on a subset of the TIMIT database, which includes 5 phones (/sh/, /ch/, /jh/, /z/, and /s/) with frequency components around 4kHz and evaluate on the complete test set. We further perform a 4-class speech emotion recognition (happy+excited, sad, angry, neutral) test on the IEMOCAP [21] database and evaluate

Table 3: *Results for real tasks (UAR%).*

TIMIT	# Filters N_1			
Pooling Size	2	4	8	16
No Pooling	73.1	76.0	75.7	75.5
2	70.0	74.4	74.7	75.4
4	70.0	72.6	74.3	73.5
8	68.6	72.2	73.6	73.9
IEMOCAP				
No Pooling	56.1	55.9	55.4	55.4
2	54.7	54.7	54.7	55.0
4	55.5	55.3	55.4	54.8
8	55.5	55.3	55.7	55.4

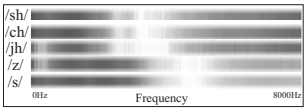

Figure 9: *Heatmap of frame-level spectra of the TIMIT subset.*

using a five-fold leaving-one-session-out strategy, i.e., using a similar experimental setting as in [4]. The classes are not balanced in both tests. Hence, we use random oversampling for training and use unweighted average recall (UAR) as the test metric. The input is padded or cut to 0.2s and 6s in the two experiments, respectively. We repeat the experiment with different pooling sizes K and numbers of filters in the first convolutional layer N_1. As shown in Table 3, for the TIMIT test, we find that no pooling and smaller pooling sizes lead to noticeable better performance and that the performance gap narrows when the first convolutional layer has a sufficient number of filters. Both phenomena match our analysis and results in experiment 3, demonstrating that the compression of pooling does affect the performance. Considering that most frequency components in the selected subset fall around the aliasing region (shown in Figure 9), this task is indeed easily affected by the aliasing effect. In contrast, the performance difference in the emotion recognition task is minor, indicating that not all real tasks will be noticeably affected by the pooling compression. It is worth mentioning that although eliminating the pooling can avoid aliasing (as shown in experiment 2), it might also hurt the performance, since the pooling layer can increase the robustness of the DNNs to slight temporal distortions in the input signal [11], hence the result might underestimate the impact of the aliasing effect.

3. Conclusions

This paper explores the impact of aliasing led by pooling in temporal CNNs. We show that combining the activation function and max/average pooling leads to aliasing, but with a smaller impact range in the frequency domain. This leads to a performance drop in designed and real audio recognition tasks, since the preceding convolutional layers do not effectively learn filterbanks able to separate the mirrored signal components. By eliminating the pooling layer, the model can eliminate the aliasing problem and obtain better performance. But this is not a practical solution, since it will greatly increase the memory and computational overhead, preventing deeper networks. Possible future research directions are the design of new activation functions and pooling techniques that can make mirrored signals distinguishable and the design of initializers, regularizers, and training schemes that help DNNs to learn anti-aliasing filters.

4. References

[1] F. Eyben, M. Wöllmer, and B. Schuller, "Opensmile: the munich versatile and fast open-source audio feature extractor," in *Proceedings of the 18th ACM international conference on Multimedia*. ACM, 2010, pp. 1459–1462.

[2] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[3] W. Zheng, J. Yu, and Y. Zou, "An experimental study of speech emotion recognition based on deep convolutional neural networks," in *Affective Computing and Intelligent Interaction (ACII), 2015 International Conference on*. IEEE, 2015, pp. 827–831.

[4] S. Ghosh, E. Laksana, L.-P. Morency, and S. Scherer, "Representation learning for speech emotion recognition," *Interspeech 2016*, pp. 3603–3607, 2016.

[5] A. M. Badshah, J. Ahmad, N. Rahim, and S. W. Baik, "Speech emotion recognition from spectrograms with deep convolutional neural network," in *Platform Technology and Service (PlatCon), 2017 International Conference on*. IEEE, 2017, pp. 1–5.

[6] P. Golik, Z. Tüske, R. Schlüter, and H. Ney, "Convolutional neural networks for acoustic modeling of raw time signal in lvcsr," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

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

[8] E. Variani, T. N. Sainath, I. Shafran, and M. Bacchiani, "Complex linear projection (clp): A discriminative approach to joint feature extraction and acoustic modeling." in *INTERSPEECH*, 2016, pp. 808–812.

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

[10] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform cldnns," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

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

[12] D. Palaz, M. Magimai.-Doss, and R. Collobert, "Analysis of cnn-based speech recognition system using raw speech as input," Idiap, Tech. Rep., 2015.

[13] G. Trigeorgis, F. Ringeval, R. Brueckner, E. Marchi, M. A. Nicolaou, B. Schuller, and S. Zafeiriou, "Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5200–5204.

[14] Y. Aytar, C. Vondrick, and A. Torralba, "Soundnet: Learning sound representations from unlabeled video," in *Advances in Neural Information Processing Systems*, 2016, pp. 892–900.

[15] W. Dai, C. Dai, S. Qu, J. Li, and S. Das, "Very deep convolutional neural networks for raw waveforms," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 421–425.

[16] 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.

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

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

[19] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[20] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[21] C. Busso, M. Bulut, C.-C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J. N. Chang, S. Lee, and S. S. Narayanan, "Iemocap: Interactive emotional dyadic motion capture database," *Language resources and evaluation*, vol. 42, no. 4, p. 335, 2008.