Acoustic modelling from the signal domain using CNNs

Pegah Ghahremani[1], *Vimal Manohar*[1], *Daniel Povey*[1,2], *Sanjeev Khudanpur*[1,2]

[1]Center of Language and Speech Processing
[2]Human Language Technology Center Of Excelence,
Johns Hopkins University, Baltimore, MD
{pghahre1,vmanoha1,khudanpur}@jhu.edu, dpovey@gmail.com

Abstract

Most speech recognition systems use spectral features based on fixed filters, such as MFCC and PLP. In this paper, we show that it is possible to achieve state of the art results by making the feature extractor a part of the network and jointly optimizing it with the rest of the network. The basic approach is to start with a convolutional layer that operates on the signal (say, with a step size of 1.25 milliseconds), and aggregate the filter outputs over a portion of the time axis using a network in network architecture, and then down-sample to every 10 milliseconds for use by the rest of the network. We find that, unlike some previous work on learned feature extractors, the objective function converges as fast as for a network based on traditional features.

Because we found that iVector adaptation is less effective in this framework, we also experiment with a different adaptation method that is part of the network, where activation statistics over a medium time span (around a second) are computed at intermediate layers. We find that the resulting 'direct-from-signal' network is competitive with our state of the art networks based on conventional features with iVector adaptation.

Index Terms: raw waveform, statistic extraction layer, Network In Network nonlinearity

1. Introduction

Most conventional automatic speech recognition (ASR) systems use hand-crafted spectral and cepstral features such as MFCC [1], PLP [2] and mel filterbank. These are inspired from physiological models of the human auditory system and may not be the most appropriate for the final ASR objective of word error rate (WER) reduction. Deep neural networks (DNN) have been shown to be able to integrate the feature extraction stage with the classification stage. With DNNs, mel-like filterbanks can be learnt from the power spectrum [3]. Tuske et al. [4] proposed to use the raw time signal directly as input to a DNN. A 1-d convolution layer was later introduced to do time-domain filtering using the same shared parameters for different time-shifts of input [5, 6]. The filters learned can model the spectral envelope of speech signal corresponding to different phonemes [6]. To the best of our knowledge, [7] is the only work where a raw waveform system is shown to give a better WER than the conventional features. However, they only report results using a very large training set. It is not clear how any of these approaches would perform on standard LVCSR tasks relative to the state-of-the-art systems that includes speaker adaptation. Our current work beats the recognition performance of a state-of-the-art MFCC-based time-delay neural network (TDNN) [8] system [9] on the WSJ task and matches the performance on the Switchboard [10] task.

One of the issues with learning directly from the waveform is the large number of variations for a given phoneme in the form of phase shifts and temporal distortions. Since the input to the DNN is at a very fast rate (8-16 kHz), using a wide temporal context would result in needing a very large affine component in the case of a TDNN and difficulties in performing backpropagation through time in a recurrent neural network (RNN). A typical approach to deal with these issues is to do max-pooling over time [11]. In [7], conventional pooling approaches such as max, p-norm and averaging functions are compared. Bhargava et al. [12] uses a pre-trained bottleneck DNN to extract features that are spliced at a slower 10 ms period over a long temporal window. In this paper, we present a novel network-in-network (NIN) [13] architecture that aggregates filter outputs over time. With this architecture, the network can even train faster than our baseline MFCC-based TDNN (See figure 2).

With traditional features, speaker adaptation is usually done using a generative framework that involves transforming features to a different space using fMLLR [14] etc. or applying a speaker-dependent bias using iVectors [15, 16]. However, iVectors are not straight-forward to work with, especially in mismatched conditions [17], and requires careful pre-processing such as segmentation and additional architectural tricks [18]. We could not get iVectors working as well in the raw waveform setup as in the MFCC setup. We instead experiment with an adaptation approach that uses activation statistics of hidden units accumulated over about a 2 second long window. Our approach eliminates the performance gap compared to the MFCC-based TDNN system with iVector speaker adaptation.

The rest of the paper is organized as follows. Section 2 describes the methods we use to preprocess and perturb the raw waveform data during training and testing. Section 3 describes the CNN-based neural network architecture used in our raw waveform setup and gives a detailed description of the proposed NIN nonlinearity and statistics extraction layer. Section 4 gives an empirical analysis of our raw waveform setup using experiments conducted on standard LVCSR tasks.

2. Raw waveform processing

In this section, we describe the pre-processing on the raw waveform before supplying it as input to the neural network.

This work was partially supported by NSF CRI Grant No 1513128, DARPA LORELEI Contract No HR0011-15-2-0024 and IARPA BABEL Contract No 2012-12050800010.

2.1. Waveform normalization

The input frames to the neural network are non-overlapping 10 ms long segments of the raw waveform signal. The raw waveform samples are quantized to 16 bits per sample, and mean and variance normalized at the utterance level . The mean variance normalization can be important for stable training [4]. The raw waveform setup with no input normalization converges a bit slower in the early stage of training, but its final performance is the same as normalized system on WSJ.

2.2. Data perturbation

We augment the data by producing 3 versions of the original signal with speed factors 0.9, 1.0 and 1.1 [19] in both the raw waveform and MFCC-based setups. The Fourier modulus is translation invariant and stable to additive noise but unstable to small deformations at high frequencies [20]. FFT-based features such as MFCC and PLP are invariant to modest translations. The large amount of variations in the raw waveform input for a given phoneme can be detrimental to training. One approach to mitigate this is to artificially perturb the data to make the network invariant to those perturbations. To achieve translation invariance, we perturb the raw input signal during training by shifting the samples randomly to right for up to of 20% of the frame window size. This means that in different epochs, we might see the same data at different shifts.

Random perturbation can greatly help in improving raw waveform performance on small datasets such as WSJ. Table 1 shows the effect of random shifts on final validation and training log-likelihoods.

Table 1: *Data perturbation effect on WSJ.*

Perturbation method	Training CE	Validation CE
No random shift	-0.96	-1.22
With random shift	-0.88	-1.13

3. Neural network architecture

In this section, we explain key features of the neural network used in our raw waveform setup. We start by describing the two novel architectural additions – the network-in-network nonlinearity in section 3.1 and the statistics extraction layer in section 3.2. We finally give the overall architecture used in our CNN-based raw waveform setup in section 3.3.

3.1. Network-in-network (NIN) nonlinearity

We introduce a new type of nonlinearity that is a special case of the network-in-network nonlinearity proposed in [13]. It is a many-to-many nonlinearity consisting of two block diagonal matrices, with repeated blocks, interleaved between layers of rectified linear units (ReLU). A normalization layer [21] is always added after the NIN nonlinearity to stabilize training. Figure 1 shows a graphical representation of the nonlinearity.

The transformation block U_1 of size $m \times k$ maps an input of size m into a higher dimensional space with dimension k, and it is subsequently passed through ReLU. We will refer to the quantity k as the "*NIN hidden dimension*". The second transformation block U_2 of size $k \times n$ maps it down to a lower dimensional space with dimension n followed by another rectification using ReLU. We will refer to the combination of U_1 block and U_2 block along with the ReLUs as a "*micro neural network block*" as marked in figure 1.

To concisely describe the proposed NIN nonlinearity, we can say that it is a group of *micro neural network blocks* applied to non-overlapping patches of input with each block being a nonlinear transformation from m dimensional space to n dimensional space.

If the *micro neural network block* parameters are shared across the NIN, each column of the block U_1 can be interpreted as a 1-d convolution filter with a filter size m and a filter shift m. Thus, the same filter is applied to non-overlapping patches and this models local connectivity. The shared parameters in the NIN nonlinearity keeps its total parameter count low relative to the size of its input and output, and allows it to be trained faster.

Figure 2 compares the training log-likelihoods for raw waveform setup using NIN nonlinearity against the baseline MFCC system. The raw waveform setup with proposed nonlinearity converges faster than the best baseline MFCC setup and the final objective value is better than baseline.

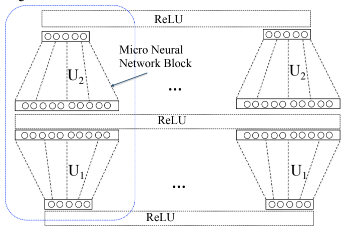

Figure 1: *Proposed NIN nonlinearity*

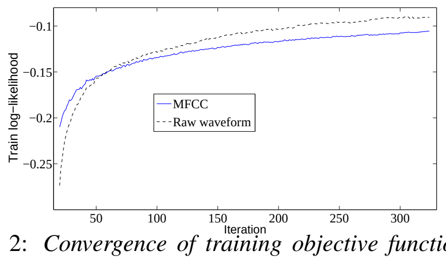

Figure 2: *Convergence of training objective function in raw waveform setup using NIN nonlinearity vs MFCC setup using ReLU.*

3.2. Statistics extraction layer

This layer extracts 1st and 2nd order statistics from hidden layer activations. These statistics are computed over a moving window of up to 200 frames (2 seconds) and appended to the input of the next hidden layer.

Given an n-dimensional input, this layer computes an $2n$ dimensional output consisting of the moving average mean of the activations and the raw diagonal 2nd-order statistics.

We expect this layer to capture long term effects in the signal such as speaker, channel and environment characteristics. This is particularly useful in the raw waveform setup as the raw signal has more information related to these characteristics, which are not in MFCCs.

3.3. CNN-based raw waveform setup

Our raw waveform setup consists of two parts – a feature extraction block and a classification block. The feature extraction block described in Section 3.3.1 consists of a CNN layer to process the raw waveform samples. The CNN outputs are aggregated using the proposed NIN nonlinearity. The classification block in our setup uses the basic TDNN architecture, but using the proposed NIN as the nonlinearity instead of ReLU. This is described in detail in Section 3.3.2.

3.3.1. Feature extraction block

The feature extraction block is illustrated in figure 3. It has a 1-d convolution layer, which takes M samples of the raw waveform and convolves them with N K-dimensional filters. S is the step size taken along input axis for computing the next dot-product of input and filters. Using the step size is equivalent to subsampling the output of convolution by a rate S. This helps in reducing computation time. The output dimension of convolution layer is $N \times D$, where $D = \frac{M-K}{S} + 1$. Next, we take the absolute value of the filter outputs and take the logarithm.

The major difference in our raw waveform architecture compared to the CNN-based setups in [11, 7, 22] is the use of a NIN nonlinearity. These setups use conventional pooling techniques such as max pooling over time to reduce the output of the convolutional layer to $N \times 1$ from $N \times D$. Our proposed NIN nonlineariy (see Section 3.1) takes the place of this pooling layer. The U_1 block of the first NIN nonlinearity is chosen to be of dimension $D \times k$, where k is the NIN hidden dimension that is typically around $5D$. A single *micro neural network block* is shared across all N filters. Each *micro neural network block* aggregates information over D samples. We have two consecutive layers of the proposed NIN nonlinearity. The normalization layer that we use after each nonlinearity scales down the output and keeps its averaged norm in check.

In speaker adaptaion experiments using raw waveform, iVectors are appended to the NIN output at this stage after first being passed through a separate affine component and a ReLU layer [18]. In the MFCC setup, we append and pass the iVectors and MFCC through an LDA transformation [21].

Figure 5 compares training convergence rates using 3 different approaches to do pooling over time in feature extraction block, while keeping the rest of the network the same. As shown, using NIN to aggregate outputs converges faster than using both p-norm pooling and no pooling. Our experiments also shows that the NIN aggregation gives 1% WER improvment over using p-norm pooling.

3.3.2. Classification block

Figure 4 shows the structure of a DNN layer used in the classification block. We use a TDNN architecture [23] to splice D_3 dimensional inputs at different time steps $t_1, t_2, \cdots, t_n$, but also append to this the moving statistics extracted using the statistics extraction layer (Section 3.2). Then, we rearrange the dimensions of the spliced input so that the L shared *micro neural network blocks* in the NIN nonlinearity is applied on $d = \frac{D_3}{L}$ dimensional patches of input data extracted from all the n time steps appended with the statistics extracted over time steps $t_l, \cdots, t_r$ for the same d dimensions. The NIN nonlinearity is followed by a normalization layer and a full affine transform to reduce the output dimension to D_3. We stack several layers of this type to form the classification block.

4. Results

4.1. Experimental setup

We conduct our experiments on two corpora – 300 hours switchboard conversational telephone speech corpus [10] and $\sim$ 80 hours Wall Street Journal continuous speech corpus [24]. All our experiments are conducted using the Kaldi Speech Recognition Toolkit [25] For the baseline, we use DNNs in time-delay neural network architecture with 40-dim MFCC features as input. For speaker-adapted systems, 100-dim iVectors

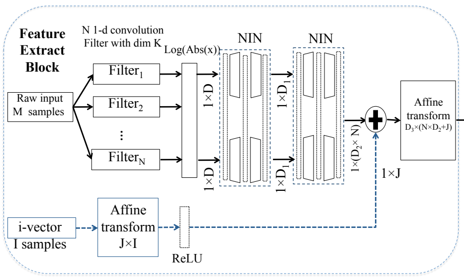

Figure 3: *Raw waveform feature extraction Block.*

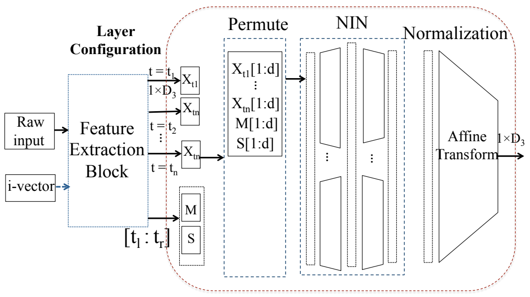

Figure 4: *Layer configuration in raw waveform classification block.*

are appended to the input features. The reader is directed to [9] for the architectural details.

4.2. Convolution filter analysis

The reasonable results using our raw waveform setup show that the first layer of network learns useful band-pass filters. The filters learned by our network are similar to those observed in other raw waveform works in the literature [5, 7, 22]. Figure 6 shows the learned filters in the form of a time-domain amplitude plot as well as a magnitude frequency response plot. The filters have been sorted by the center frequencies. The frequency-domain plot shows that the center frequencies of the filters increases linearly until 1000 Hz, which is very similar to mel filterbanks.

4.3. Results

4.3.1. WSJ task

The raw waveform setup used in WSJ experiments is as described in Section 3, we use both p-norm pooling and NIN nonlinearity in the feature extraction block and conventional TDNN layer used in the classification block. The networks here, including the MFCC baselines, are trained using frame cross-entropy objective. The statistics extraction layer is not used in these experiments as the cross-entropy model trains on small chunks over which we cannot extract reliable statistics.

The CNN layer in feature extraction block consists of 40 filters. The filter size 30 ms is used on 50 ms raw waveform signal that is sampled at 16 kHz and the filter step size used is 0.62 ms. Table 2 compares results of our raw waveform setup and the MFCC-based TDNN system on the WSJ 5K vocabulary task. First 3 rows are the systems without speaker adaptation. In 2^{nd} experiment, the output of the convolution filters are pooled over time using p-norm pooling and in 3^{rd} and last experiment, the NIN nonlinearity with 40 *micro neural network blocks* with input size 16, NIN hidden dimension 300 and output size 32

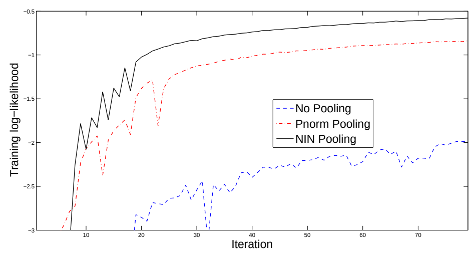

Figure 5: *Training log likelihood for different pooling method.*

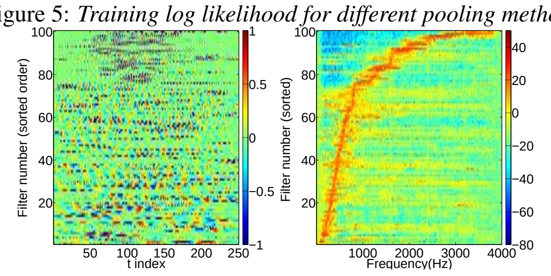

Figure 6: *Amplitude plot (top) and magnitude response (bottom) of learned filters ordered by center frequency*

are used. The classification block has 6 hidden layers, each with 750 ReLU units. We see that the raw waveform system performs more than 1% absolute better than the MFCC baseline.

From the next two lines, we see that adding iVectors to the MFCC system improves the MFCC baseline but still does not beat the raw waveform setup without iVectors. This may indicate that our raw waveform setup is less sensitive to speaker mismatches. Adding iVectors to the raw waveform setup actually degrades the result. In the final experiment, we tried adding the NIN nonlinearity, but could not get any improvment over ReLU.

Table 2: *WER (%) Results on WSJ LVCSR task.*

Model	*Nov'92 eval*	*Nov'93 dev*
MFCC	5.28	8.29
Raw	3.95	7.34
Raw + NIN	3.92	7.6
MFCC + iVector	4.52	7.51
Raw + iVector	4.06	7.80

4.3.2. Switchboard task

Table 4 compares results of the proposed raw waveform system and the MFCC-based TDNN system on the switchboard task. The results are reported both the *Hub5'00* evalutation set and the *RT'03* evalutation set.

In raw waveform setup, the CNN layer consists of 100 filters. The filter size used is 31.25 ms on a 50 ms raw waveform signal that is sampled at 8 kHz and the filter step size is 1.25 ms. The feature extraction block uses NIN nonlinearity with 100 *micro neural network blocks* with input size $m = 16$ (same as convolution filter output dim), NIN hidden dimension $k = 120$ and output size $n = 18$. The output dimension of the feature extraction block is $D_3 = 500$. The classification block has 6 hidden layers, with either ReLU nonlinearity (600 hidden units) or NIN nonlinearity. The NIN nonlinearity has 100 *micro neural network blocks* with input size $m = 5$, NIN hidden dimension $k = 75$ and output size $n = 18$. The neural networks are trained using lattice-free MMI [9]. In the experiments using the statistics extraction layer, mean and standard deviation of the hidden layer activations are computed over the available frames on either side for up to a maximum of 99 frames on either side.

Table 3 shows the effect of using NIN nonlinearity in the classification block in the raw waveform setup. The ReLU system and the NIN system, both have the same TDNN structure in terms of context [23] and use the same feature extraction block including iVectors. We see that using NIN nonlinearity gives 1% improvement over the conventional ReLU nonlinearity in the raw waveform setup. However, we found that the NIN nonlinearity does not give any improvement over ReLU on the MFCC setup.

Table 3: *Effect of NIN nonlinearity*

	Hub5'00		*RT'03*	
Model	Total	SWBD	Total	SWBD
ReLU	17.2	11.5	19.9	24.0
NIN	16.1	10.5	18.9	23.1

Table 4 compares the raw waveform setup with NIN nonlinearity and MFCC setup with ReLU nonlinearity. The first two rows in the table are the results without speaker adaptation.

The next two rows show the effect of adding the statistics extraction layer. We see that statistics extraction layer improves the performance of both MFCC and raw waveform setups. The raw waveform setup works slightly better than the MFCC setup, which may indicate the statistic layer helps the network to extracts some speaker or channel dependent information directly from raw waveform, which may removed during MFCC extraction process.

The last two rows show the effect of doing speaker adaptation by adding iVectors. We see that iVectors give a lot of improvement in the MFCC setup, but only a little improvment in the raw waveform setup. We hypothesize that the raw waveform possesses more information than the MFCC features and the network can learn to account for the speaker and environment variability. However, we need to do more experiments to verify this.

Table 4: *WER (%) Results on Switchboard LVCSR task.*

	Hub5'00		*RT'03*	
Model	Total	SWBD	Total	SWBD
MFCC	17.5	11.6	22.1	26.6
Raw	17.4	11.5	21.7	26.5
MFCC + Stats	16.4	11.0	20.0	24.3
Raw + Stats	16.3	10.6	19.1	23.3
MFCC + iVector	15.7	10.4	19.2	23.5
Raw + iVector	16.1	10.5	18.9	23.1

5. Conclusion

We proposed a CNN-TDNN based architecture for the raw waveform setup including a repeated "network-in-network structure which aggregates the time information from convolution filter outputs. We show that results are improved if this NIN nonlinearity takes the place of the conventional ReLU hidden layer in the DNN. We were not able to get improvement from adding iVectors to the network using raw-waveform inputs, but we got almost as much improvement from a different adaptation method internal to the network. It extracts mean and standard deviations from moving-average windows of hidden-layer activations. On the Switchboard task that we can compete with even state-of-the-art iVector speaker-adapted MFCC systems; thus eliminating the need for iVector based speaker adaptation. The performance of the proposed raw waveform setup needs to be investigated on other datasets, including in noisy and mismatched conditions.

6. References

[1] P. Mermelstein, "Distance measures for speech recognition, psychological and instrumental," *Pattern recognition and artificial intelligence*, vol. 116, pp. 374–388, 1976.

[2] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *Journal of the Acoustical Society of America*, vol. 87, pp. 1738–1752, 1990.

[3] T. N. Sainath, B. Kingsbury, A.-r. Mohamed, and B. Ramabhadran, "Learning filter banks within a deep neural network framework," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 297–302.

[4] Z. Tüske, P. Golik, R. Schlüter, and H. Ney, "Acoustic modeling with deep neural networks using raw time signal for LVCSR," in *Proc. Interspeech*, 2014.

[5] 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.

[6] D. Palaz, R. Collobert *et al.*, "Analysis of CNN-based Speech Recognition System using Raw Speech as Input," in *Proceedings of Interspeech*, no. EPFL-CONF-210029, 2015.

[7] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform cldnns," in *Proc. Interspeech*, 2015.

[8] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, Mar 1989.

[9] D. Povey, V. Peddinti, D. Galvez, P. Ghahrmani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for ASR based on lattice-free MMI," in *Submitted to Interspeech*, 2016. [Online]. Available: http://www.danielpovey.com/files/2016_interspeech_mmi.pdf

[10] J. J. Godfrey *et al.*, "Switchboard: Telephone speech corpus for research and development," in *ICASSP*, 1992.

[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] M. Bhargava and R. Rose, "Architectures for deep neural network based acoustic models defined over windowed speech waveforms," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

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

[14] M. J. F. Gales and P. C. Woodland, "Mean and Variance Adaptation Within the MLLR Framework," *Computer Speech and Language*, vol. 10, pp. 249–264, 1996.

[15] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 4, pp. 788–798, 2011.

[16] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors." in *ASRU*, 2013, pp. 55–59.

[17] V. Peddinti, G. Chen, V. Manohar, T. Ko, D. Povey, and S. Khudanpur, "JHU ASpIRE system: Robust LVCSR with TDNNs, i-vector Adaptation, and RNN-LMs," in *ASRU*, 2015.

[18] S. Garimella, A. Mandal, N. Strom, B. Hoffmeister, S. Matsoukas, and S. H. K. Parthasarathi, "Robust i-vector based adaptation of DNN acoustic model for speech recognition," *In Proceedings of Interspeech*, 2015.

[19] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Proceedings of INTERSPEECH*, 2015.

[20] J. Andén and S. Mallat, "Deep scattering spectrum," *Signal Processing, IEEE Transactions on*, vol. 62, no. 16, pp. 4114–4128, 2014.

[21] X. Zhang, J. Trmal, D. Povey, and S. Khudanpur, "Improving Deep Neural Network Acoustic Models using Generalized Maxout Networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, May 2014, pp. 215–219.

[22] 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.

[23] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proceedings of INTERSPEECH*, 2015.

[24] D. B. Paul and J. M. Baker, "The design for the Wall Street Journal-based CSR corpus," in *Proceedings of the workshop on Speech and Natural Language*. Association for Computational Linguistics, 1992, pp. 357–362.

[25] D. Povey, A. Ghoshal *et al.*, "The Kaldi Speech Recognition Toolkit," in *Proc. ASRU*, 2011.