INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

Acoustic Modeling for Google Home

Bo Li, Tara N. Sainath, Arun Narayanan, Joe Caroselli, Michiel Bacchiani, Ananya Misra, Izhak Shafran, Hasim Sak, Golan Pundak, Kean Chin, Khe Chai Sim, Ron J. Weiss, Kevin W. Wilson, Ehsan Variani, Chanwoo Kim, Olivier Siohan, Mitchel Weintraub, Erik McDermott, Richard Rose, Matt Shannon

Google, Inc. U.S.A

{boboli, tsainath, arunnt, jcarosel, michiel, amisra, izhak}@google.com
{hasim, golan, kkchin, khechai, ronw, kwwilson, variani, chanwcom}@google.com
{siohan, mweintraub, erikmcd, rickrose, mattshannon}@google.com

Abstract

This paper describes the technical and system building advances made to the Google Home multichannel speech recognition system, which was launched in November 2016. Technical advances include an adaptive dereverberation frontend, the use of neural network models that do multichannel processing jointly with acoustic modeling, and Grid-LSTMs to model frequency variations. On the system level, improvements include adapting the model using Google Home specific data. We present results on a variety of multichannel sets. The combination of technical and system advances result in a reduction of WER of 8-28% relative compared to the current production system.

1. Introduction

Farfield speech recognition has made great strides in the past few years, from research focused activities such as the CHiME Challenge [1] to the launch of Amazon Echo and Google Home. Farfield speech recognition is challenging since the speech signal can be degraded by reverberation and additive noises, significantly degrading word error rate (WER). Such systems are not usable until the WER comes into a manageable range. The purpose of this paper is to detail the technical and system advances in acoustic modeling that have gone into the Google Home system.

A typical approach to farfield recognition is to use multiple microphones to enhance the speech signal and reduce the impact of reverberation and noise [2, 3, 4]. While multichannel ASR systems often use separate models to perform beamforming and acoustic modeling, we recently proposed doing this jointly in a neural network using raw-waveforms as input [5]. In [5], we explored a variety of architectures in both the time-domain and frequency domain. Taking into account the trade-offs between computational complexity and performance, we propose to use the factored Complex Linear Projection (fCLP) model [6] in the current work, which has much smaller computational complexity and similar performance to models trained in the time domain. We will also show in the current work that doing multichannel processing jointly with acoustic modeling is better compared to an acoustic model trained with log-mel features as the latter has limited ability to do spatial processing.

The fCLP model takes a complex fast Fourier transform (CFFT) as input, and mimics filter-and-sum operations in the first layer. To further improve robustness of such models, we explored a dereverberation feature frontend. Specifically, a mutlichannel recursive least squares (RLS) adaptive algorithm is applied [7]. This algorithm is based on the weighted prediction error (WPE) algorithm [8]. It reduces the effects of reveberation, thereby helping the neural network process multichannel input more effectively.

We also improve the acoustic model using a Grid-LSTM [9]. Recently, we have observed that Grid-LSTMs are able to better model frequency variations, particularly in noisy conditions, compared to a convolutional layer [10]. In this work, the output of the fCLP layers, which closely resembles a time-frequency feature in different look directions, is passed to a Grid-LSTM.

Our experiments to understand the benefit of the different modules are conducted on a 18,000 hr Voice Search task. We find that the fCLP layer provides up to 7% relative improvement in noisy conditions over an acoustic model trained with log-mel features. Including WPE results in an additional 7% improvement in the noisiest conditions, while the Grid-LSTM improves performance by 7-11% relative in all conditions. By combining all of these technical improvements, we obtain an overall improvement of 16% compared to the existing log-mel production system on an evaluation set collected from the Google Home traffic. Finally, adapting the acoustic model via sequence training on approximately 4,000 hours of training data collected from live traffic improves WER by 8-28% relative over the baseline.

Overall, with both the technical and system advances, we are able to reduce the WER to 4.9% absolute, a 20% relative reduction over a log-mel trained LSTM CTC model.

The rest of this paper is as follows. In Section 2 we highlight the overall architecture explored in this paper, with the WPE, fCLP and Grid-LSTM submodules to be discussed in Sections 3, 4 and 5, respectively. The experimental setup is described in Section 6, while results are presented in Section 7. Finally, Section 8 concludes the paper and discusses future work.

2. System Overview

A block diagram of the proposed system is shown in Figure 1. The CFFT for each channel is first passed to an adaptive WPE frontend that performs dereverberation. The WPE processed CFFT features are fed to a fCLP layer, which does multichannel processing and produces a time-frequency representation. The output of fCLP processing is passed to a Grid-LSTM to model time and frequency variations. Finally, the output of the Grid-LSTM goes to a standard LDNN acoustic model [11]. The WPE, fCLP and Grid-LSTM modules will be discussed in the next three sections.

http://dx.doi.org/10.21437/Interspeech.2017-234

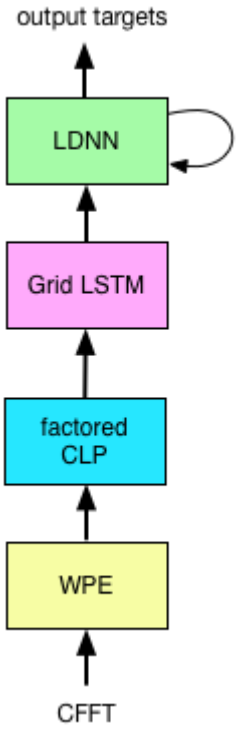

Figure 1: *System Overview of Google HOME*

3. Dereverberation

Reverberation is often modeled as a convolution of the clean speech signal with a room impulse response (RIR). This convolution introduces correlation in time in the speech that would not otherwise be present. Dereverberation can be performed by estimating this additional correlation and filtering to undo it.

The weighted prediction error (WPE) algorithm [8] is one such technique that has shown promising results [4, 12]. WPE requires that the entire utterance to be obtained before the filter taps can be calculated and, consequently, before dereverberation can be performed. For applications that need streaming recognition, like Google Home, this latency is not acceptable. The filter coefficients must be estimated and applied as quickly as the speech signal arrives. Furthermore, it is desirable for the tap values to be adaptable because the RIRs will change due to reasons like speaker motion or because of the nonstationarity of the speech signal itself.

A single channel RLS-based dereverberation algorithm was presented in [13]. A variation of this algorithm that extends to the multi-channel case is presented in [7] and applied here.

The adaptive algorithm is applied in the frequency domain. The FFT size was selected considering the coherence bandwidth of typical RIRs such that the channel response of adjacent frequency bins is roughly uncorrelated. The tap values are calculated independently for each frequency bin.

Let the vector $\mathbf{Y}_l[n]^T \equiv [\mathbf{Y}_{0,l}[n] \quad \mathbf{Y}_{1,l}[n]]$ represent frame n and frequency bin l of the STFT of the received signal for each of the two microphones. A vector of N delayed STFT frames from each of the 2 microphones is given by $\tilde{\mathbf{Y}}_l[n]^T \equiv \left[\tilde{\mathbf{Y}}_{0,l}^T \quad \tilde{\mathbf{Y}}_{1,l}^T\right]$, where

$$\tilde{\mathbf{Y}}_{m,l}[n]^T \equiv \left[\mathbf{Y}_{m,l}[n-\Delta] \quad \cdots \quad \mathbf{Y}_{m,l}[n-\Delta-N+1]\right]. \tag{1}$$

This vector of delayed samples is passed through parallel filters represented by the $2N \times 2$ matrix $\mathbf{W}_l[n]$. The filter outputs are subtracted from the vector of received signals to produce the dereverberated output $\hat{\mathbf{Y}}_l[n]$ as shown below:

$$\hat{\mathbf{Y}}_l[n] = \mathbf{Y}_l[n] - \mathbf{W}_l[n]^H \tilde{\mathbf{Y}}_1[n]. \tag{2}$$

Equation (2) is applied at every frame for each frequency bin.

The taps are updated at each time step according to the Kalman filter update equation:

$$\mathbf{W}_l[n] = \mathbf{W}_l[n-1] + \mathbf{K}_l[n]\hat{\mathbf{Y}}_l^H[n], \tag{3}$$

where the Kalman gain is given by:

$$\mathbf{K}_l[n] \equiv \frac{\mathbf{R}_{\tilde{y}\tilde{y},l}^{-1}[n-1]\tilde{\mathbf{Y}}[n]}{\alpha\hat{\Lambda}_l^2[n] + \tilde{\mathbf{Y}}[n]^H\mathbf{R}_{\tilde{y}\tilde{y},l}^{-1}[n-1]\tilde{\mathbf{Y}}[n]} \tag{4}$$

and:

$$\mathbf{R}_{\tilde{y}\tilde{y},l}[n] \equiv \sum_{k=0}^{n} \frac{\alpha^{n-k}}{\hat{\Lambda}_l^2[k]} \tilde{\mathbf{Y}}_i[k]\tilde{\mathbf{Y}}_i[k]^H \tag{5}$$

is the weighted autocorrelation in time of the delayed data in frequency bin l. α is a forgetting factor ($0 < \alpha \leq 1$) that impacts adaptation speed. $\hat{\Lambda}_l^2[n]$ is an estimate of the signal power of frequency bin l and frame n.

4. Multichannel Processing from Complex Spectra

After WPE is applied to the CFFT, it is passed to a fCLP layer. The architecture of the fCLP layer that we use follows our earlier work described in [6]: The first layer of the model, also called the factoring layer, mimics multiple filter-and-sum operations, and is followed by complex linear projection (CLP). The system in [6] operates at 100 Hz input frame rate. The input to the network is the multichannel complex spectra computed for a window of 32 millisecond with a frame-shift of 10 millisecond. Results in [14] [15] show that for logmel features we can reduce the frame rate by a factor of 3 to improve both performance and decoding speed. We extend this to CFFT models by reducing the frame rate in two ways: 1) Weight sharing (WS) or 2) Autoregressive filtering (AR).

In weight sharing, we continue to operate the CFFT layers at 100 Hz as in [6]:

$$Y^p[n] = \sum_{c=1}^{C} X_c[n] \cdot H_c^p, \tag{6}$$

$$Z_f^p[n] = \log\left|\sum_l Y^p[n,l] \cdot G_f[l]\right| \tag{7}$$

Here, n indexes frames (at 100 Hz), c indexes microphone channel, l indexes FFT bins, and f indexes CLP filters. X_c is the input frame for channel c, H_c^p is one of the P complex filters for channel c that mimics filter-and-sum operation, and G_f is one of the F CLP filters. In contrast to [6], the output activations, $\{Z_f^p[n]$ for $f \in 1 \ldots F\}$, are stacked across both $p \in \{1 \ldots P\}$ and $n \in \{T_l \ldots T_h\}$ where, T_l and T_h define a local temporal context. They are set to -3 and 1, respectively, in our experiments. The stacked activations are then subsampled by a factor of 3 to reduce the frame rate. The LSTMs above the CFFT layers operate at 33 Hz.

When using autoregressive filtering, we stack input features and constrain the CFFT layers to learn filters that span a much wider context. Mathematically, the factoring component of the CFFT layer is redefined as:

$$Y^p[3n] = \sum_{t=-T_l}^{T_h} \sum_{c=1}^{C} X_c[3n+t] \cdot H_{c,t}^p \quad (8)$$

Here, $t \in T_l \dots T_h$ denotes the time index of AR filter, $H_{c,t}^p$ is the complex filter for the pth look direction of the factoring layer for channel c and AR filter context t. The advantage of using an AR formulation is that the network could potentially learn spatial filters with a longer timespan. The output of the factoring layer is subsampled to 33 Hz and is passed to the CLP layer. Unlike the WS approach, the output of the CLP layer need to be stacked only across the P look directions.

5. Grid-LSTMs

The output of the fCLP layer is passed to a Grid-LSTM [9], which is a type of two dimensional LSTM that uses separate LSTMs to model the variations across time and frequency [10]. However, at each time-frequency bin, the grid frequency LSTM (gF-LSTM) uses the state of the grid time LSTM (gT-LSTM) from the previous timestep, and similarly the gT-LSTM uses the state of the gF-LSTM from the previous frequency step. The motivation for looking at the Grid-LSTM is to explore benefits of having separate LSTMs to model the correlations in time and frequency. In this work, a bidirectional Grid-LSTM [16] is adopted. It utilizes a bidirectional LSTM in the frequency direction to mitigate the directional dependency incurred by the unidirectional LSTM. While for the time direction, we keep the unidirectional LSTM. This way we can maintain the capability of processing the speech signal in an online fashion. Furthermore in [16], we have found the use of bidirectional frequency processing allows us to use non-overlapping filters which actually reduces the computation costs by a lot.

The bidirectional Grid-LSTM consists of a forward Grid-LSTM and a backward Grid-LSTM. The forward processing ('fwd') consists of following steps with $u \in \{i, f, c, o\}$:

$$q_{u,t,k}^{(\texttt{fwd})} = W_{um}^{(\texttt{fwd},t)} m_{t-1,k}^{(\texttt{fwd},t)} + W_{um}^{(\texttt{fwd},k)} m_{t,k-1}^{(\texttt{fwd},k)} \quad (9)$$

$$i_{t,k}^{(\texttt{fwd},s)} = \sigma(W_{ix}^{(\texttt{fwd},s)} x_{t,k} + q_{i,t,k}^{(\texttt{fwd})} + b_i^{(\texttt{fwd},s)}) \quad (10)$$

$$f_{t,k}^{(\texttt{fwd},s)} = \sigma(W_{fx}^{(\texttt{fwd},s)} x_{t,k} + q_{f,t,k}^{(\texttt{fwd})} + b_f^{(\texttt{fwd},s)}) \quad (11)$$

$$c_{t,k}^{(\texttt{fwd},t)} = f_{t,k}^{(\texttt{fwd},t)} \odot c_{t-1,k}^{(\texttt{fwd},t)} + i_{t,k}^{(\texttt{fwd},t)} \odot g(W_{cx}^{(\texttt{fwd},t)} x_{t,k} + q_{c,t,k}^{(\texttt{fwd})} + b_c^{(\texttt{fwd},t)}) \quad (12)$$

$$c_{t,k}^{(\texttt{fwd},k)} = f_{t,k}^{(\texttt{fwd},k)} \odot c_{t,k-1}^{(\texttt{fwd},k)} + i_{t,k}^{(\texttt{fwd},k)} \odot g(W_{cx}^{(\texttt{fwd},k)} x_{t,k} + q_{c,t,k}^{(\texttt{fwd})} + b_c^{(\texttt{fwd},k)}) \quad (13)$$

$$o_{t,k}^{(\texttt{fwd},s)} = \sigma(W_{ox}^{(\texttt{fwd},s)} x_{t,k} + q_{o,t,k}^{(\texttt{fwd})} + b_o^{(\texttt{fwd},s)}) \quad (14)$$

$$m_{t,k}^{(\texttt{fwd},s)} = o_{t,k}^{(\texttt{fwd},s)} \odot h(c_{t,k}^{(\texttt{fwd},s)}) \quad (15)$$

For the backward processing ('bwd'), we have a separate set of weights parameters $W_{**}^{(\texttt{bwd},*)}$ and $b_*^{(\texttt{bwd},*)}$. In the above equations, instead of using the previous frequency block's, *i.e.* $(k-1)$-th LSTM output $m_{t,k-1}^{(\texttt{bwd},k)}$ and cell state $c_{t,k-1}^{(\texttt{bwd},k)}$, the next frequency block's, *i.e.* $(k+1)$-th LSTM output $m_{t,k+1}^{(\texttt{bwd},k)}$ and cell state $c_{t,k+1}^{(\texttt{bwd},k)}$ are used.

The final output at each time-frequency block (t, k) is a concatenation of the forward and backward activations:

$$m_{t,k}^{(s)} = [m_{t,k}^{(\texttt{fwd},s)T} \quad m_{t,k}^{(\texttt{bwd},s)T}]^T \quad (16)$$

At each time step t, we concatenate the Grid-LSTM cell output $m_{t,k}^{(s)}$ for all the frequency block k and give them to a linear dimensionality reduction layer, followed by an LDNN.

6. Experimental Details

6.1. Corpora

We conduct experiments on about 18,000 hours of noisy training data consisting of 22 million English utterances. This data set is created by artificially corrupting clean utterances using a room simulator to add varying degrees of noise and reverberation. The clean utterances are anonymized and hand-transcribed voice search queries, and are representative of Google's voice search traffic. Noise signals, which include music and ambient noise sampled from YouTube and recordings of "daily life" environments, are added to the clean utterances at SNRs ranging from 0 to 30 dB, with an average SNR of 11 dB. Reverberation is simulated using the image model [17] – room dimensions and microphone array positions are randomly sampled from 3 million possible room configurations with RT_{60}s ranging from 0 to 900 ms, with an average RT_{60} of 500 ms. The simulation uses a 2-channel linear microphone array, with inter-microphone spacing of 71 millimeters. Both noise and target speaker locations change between utterances; the distance between the sound source and the microphone array varies between 1 to 8 meters.

We evaluate our models using simulated, rerecorded and real noisy farfield data. For the simulated and rerecorded sets, around 15 hours (13K utterances) of anonymized voicesearch utterances were used. For the simulated sets, noise is added using the room simulator with a room configuration distribution that approximately matches the training configurations. The noise snippets and room configurations do not overlap with training. For the rerecorded sets, we played the clean eval set and noise recordings separately in a living room setting (approximately 200ms RT_{60}), and mixed them artificially at SNRs ranging from 0 dB to 20 dB.

For the real farfield sets, we sample anonymized and hand-transcribed queries directed towards Google Home. The set consists of approximately 22,000 utterances, and are typically at a higher SNR compared to artificially created sets. We also present WER breakdown under various noise conditions in Section 7.

6.2. Architecture

All experiments in this paper use CFFT or log-mel features computed with a 32-ms window and shifted every 10ms. Low frame rate (LFR) [15] models are used, where at the current frame t, these features are stacked with 3 frames to the left and downsampled to a 30ms frame rate. An LDNN architecture [11] is consistent in all experiments, and consists of 4 LSTM layers with 1,024 cells/layer unless otherwise indicated, and a DNN layer with 1,024 hidden units. During training, the network is unrolled for 20 time steps for training with truncated backpropagation through time. In addition, the output state label is delayed by 5 frames, as we have observed that information about future frames improves the prediction of the current frame [15]. All neural networks are trained with the cross-entropy criterion,

using asynchronous stochastic gradient descent (ASGD) optimization [18].

7. Results

7.1. fCLP

Our first set of experiments compare log-mel and fCLP methods. For these experiments, 832 cells/layer were used in the time LSTM as the experiments ran quicker. First, Table 1 shows that the fCLP,WS method outperforms the fCLP,AR method, showing that it is beneficial to give more freedom to each look direction with the WS method. In addition, the factored CLP,WS layer gives up to 7% relative improvement over the log-mel system. While our previous result had shown the benefit of the fCLP layer over log-mel for simulated data [6], the table confirms the benefits on the rerecorded data as well. The remainder of the experiments will be conduced with the fCLP,WS layer, and for simplicity it will be referred to as fCLP.

Table 1: *WER of fCLP.*

Model	clean	simulated noisy	rerecorded	rerecorded noisy
log-mel	12.5	20.0	20.0	32.8
fCLP, AR	12.7	19.1	21.5	32.9
fCLP, WS	12.3	18.6	20.2	31.6

7.2. Grid-LSTM

We further add in the bidirectional Grid-LSTM layer for better modeling of the time-frequency correlation of speech signals. We used 128D LSTM cell states to track the changes across time and frequency separately. For the frequency processing, filters of size 16 and stride 16 are used. This configuration was found to work well. Again, 832 cells/layers are used for the LSTM and no WPE is used in these experiments. From Table 2, Grid-LSTM layer consistently improves the recognition performance across all the test sets. Especially for the noisy sets, a relative 7-11% WER reduction are obtained.

Table 2: *WER of the recognition system with (w) and without (w/o) Grid-LSTM layer in between the fCLP layer and the LDNN stack.*

Model	clean	simulated noisy	rerecorded	rerecorded noisy
w/o	12.3	18.6	20.2	31.6
w	11.4	16.6	18.3	28.9

7.3. WPE

Table 3 shows the performance with and without dereverberation for the fCLP. For speed purposes, these experiments were conducted without a Grid-LSTM layer, and with 1,024 LSTM cells states. For both training and evaluation, $N = 10$ taps have been applied for each frequency bin. This value proved to be a good balance between complexity and performance. The delay Δ used was 2 frames and the forgetting factor α is set to 0.9999. The tap values are all initialized to zero at the beginning of each new utterance.

The largest relative improvement, about 7%, is obtained on the *rerecorded noisy* dataset. A possible reason for this is that not only is there benefit from the dereverberation, but the dereverberation allows the implicit beamforming performed by the neural network to better suppress the noise. Also, examining the performance in the *clean* environment shows that there is no negative impact in the absence of reverberation and noise.

Table 3: *WER Impact of Derverberation.*

Model	clean	simulated noisy	rerecorded	rerecorded noisy
No Drvb	11.7	18.2	20.5	32.2
With Drvb	11.7	17.5	19.7	30.1

7.4. Adaptation

In this section, we combined WPE, fCLP and Grid-LSTM modules, and report results after sequence training. Rather than reporting results on the "rerecorded" sets, which were more for our understanding that different modules were working properly, we now report performance on the Google Home test set, which is representative of real world traffic. The first two rows in Table 4 show that the proposed system offers a 16% relative improvement compared to the existing log-mel LSTM production system. The major win comes in noisy environments, especially in speech background noise (26% WERR) and music noise (18% WERR) where we would expect beamforming and the Grid-LSTM to help more.

Next, we further adapt the proposed model by continuing sequence training with the 4,000 hours real traffic training set. The third row of Table 4 shows that adaptation gives an additional 4% relative improvement. Overall, the proposed technical and system advances provide approximately a 8-28% relative improvement over the production system.

Table 4: *WER on Google Home test set.*

Model	Full	Clean	Noise Type		
			Speech	Music	Other
prod	6.1	5.1	8.5	6.2	6.0
home	5.1	4.9	6.3	5.1	5.0
home(adapt)	4.9	4.7	6.1	4.9	4.8

8. Conclusions

In this paper, we described the various aspects of the Google Home multichannel speech recognition system. Technical achievements include a WPE to perform dereverberation, an fCLP to perform beamforming jointly with acoustic modeling, and a Grid-LSTM to model time-frequency variations. In addition, we also presented results by adapting the model based on data from real traffic. Overall, we are able to achieve a 8-28% relative reduction in WER compared to the current production system.

9. References

[1] E. Vincent, S. Watanabe, A. A. Nugraha, J. Barker, and R. Marxer, "An Analysis of Environment, Microphone and Data Simulation Mismatches in Robust Speech Recognition," *Computer Speech and Language*, 2016.

[2] M. Brandstein and D. Ward, *Microphone Arrays: Signal Processing Techniques and Applications*. Springer, 2001.

[3] J. Benesty, J. Chen, and Y. Huang, *Microphone Array Signal Processing*. Springer, 2009.

[4] M. Delcroix, T. Yoshioka, A. Ogawa, Y. Kubo, M. Fujimoto, N. Ito, K. Kinoshita, M. Espi, T. Hori, T. Nakatani, and A. Nakamura, "Linear Prediction-based Dereverberation with Advanced Speech Enhancement and Recognition Technologies for the REVERB Challenge," in *REVERB Workshop*, 2014.

[5] T. N. Sainath, R. J. Weiss, K. W. Wilson, B. Li, A. Narayanan, E. Variani, M. Bacchiani, I. Shafran, A. Senior, K. Chin, A. Misra, and C. Kim, "Multichannel Signal Processing with Deep Neural Networks for Automatic Speech Recognition," *IEEE Transactions on Speech and Language Processing*, 2016.

[6] T. N. Sainath, A. Narayanan, R. J. Weiss, K. W. Wilson, M. Bacchiani, and I. Shafran, "Improvements to Factorized Neural Network Multichannel Models," in *Proc. Interspeech*, 2016.

[7] J. Caroselli, I. Shafran, A. Narayanan, and R. Rose, "Adaptive multichannel dereverberation for automatic speech recognition," in *Proc. Interspeech*, 2017.

[8] T. Yoshioka and T. Nakatani, "Generalization of multi-channel linear prediction methods for blind MIMO impulse response shortening," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 10, pp. 2707–2720, 2012.

[9] N. Kalchbrenner, I. Danihelka, and A. Graves, "Grid Long Short-Term Memory," in *Proc. ICLR*, 2016.

[10] T. N. Sainath and B. Li, "Modeling Time-Frequency Patterns with LSTM vs. Convolutional Architectures for LVCSR Tasks," in *Proc. Interspeech*, 2016.

[11] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks," in *Proc. ICASSP*, 2015.

[12] T. Yoshioka, N. Ito, M. Delcroix, A. Ogawa, K. Kinoshita, M. Fujimoto, C. Yu, W. J. Fabian, M. Espi, T. Higuchi *et al.*, "The ntt chime-3 system: Advances in speech enhancement and recognition for mobile multi-microphone devices," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015, pp. 436–443.

[13] T. Yoshioka, H. Tachibana, T. Nakatani, and M. Miyoshi, "Adaptive dereverberation of speech signals with speaker-position change detection," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*. IEEE, 2009, pp. 3733–3736.

[14] A. Senior, H. Sak, T. N. S. F. de Chaumont Quitry, and K. Rao, "Acoustic Modelling with CD-CTC-SMBR LSTM RNNS," in *Proc. ASRU*, 2015.

[15] G. Pundak and T. N. Sainath, "Lower Frame Rate Neural Network Acoustic Models," in *Proc. Interspeech*, 2016.

[16] B. Li and T. N. Sainath, "Reducing the Computational Complexity of Two-Dimensional LSTMs," in *Proc. Interspeech*, 2017.

[17] J. B. Allen and D. A. Berkley, "Image Method for Efficiently Simulation Room-Small Acoustics," *Journal of the Acoustical Society of America*, vol. 65, no. 4, pp. 943 – 950, April 1979.

[18] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, "Large Scale Distributed Deep Networks," in *Proc. NIPS*, 2012.