USE OF MULTIPLE FRONT-ENDS AND I-VECTOR-BASED SPEAKER ADAPTATION FOR ROBUST SPEECH RECOGNITION

Md Jahangir Alam[1,2], Vishwa Gupta[1], Patrick Kenny[1], Pierre Dumouchel[2]

[1]Centre de recherche informatique de Montréal, Montréal, Canada
[2]École de technologie supérieure, Montréal, Canada

ABSTRACT

Although state-of-the-art speech recognition systems perform well in controlled environments they work poorly in realistic acoustical conditions in reverberant environments. Here, we use multiple front-ends (conventional mel-filterbank, multitaper spectrum estimation-based mel filterbank, robust mel and compressive gammachirp filterbank, iterative deconvolution-based dereverberated mel-filterbank, and maximum likelihood inverse filtering-based dereverberated mel-frequency cepstral coefficients) - based recognition systems with multi-condition training data and combined their results using ROVER (Recognizer Output Voting Error Reduction). For 2- and 8- channel tasks, to get benefit from more than one channel, we also utilize ROVER instead of the multi-microphone signal processing method. As in previous work we also apply i-vector -based speaker adaptation which was found effective. Speech recognition experiments are conducted on the REVERB challenge 2014 corpora using the Kaldi recognizer. For the 2-channel task (using full batch processing) we obtained an average word error rate (WER) of 9.0% and 23.4% on the SimData and RealData respectively. Whereas for 8-channel task on the SimData and RealData the average WERs found were 8.9% and 21.7%, respectively.

Index Terms— Speech recognition, multitaper, filterbank features, dereverberation, i-vectors, DNN.

1. INTRODUCTION

Automatic speech recognition is a key component in hands-free man-machine interaction. State-of-the-art speech recognition systems are based on statistical acoustic models which are trained in a clean and controlled environment. In many applications, speech recognition systems are deployed in reverberant environments. The speech signal can be highly distorted by this room reverberation. Consequently, the performance of speech recognition systems trained on clean data degrades severely in reverberant environments because of the mismatch between the training and the test conditions. Reverberation is a phenomenon where delayed and attenuated versions of a signal are added to itself. and typically modeled as a linear filtering of a signal in time domain. Compensation of acoustic reverberant environment is usually done by dereverberation, which can be obtained by inverse filtering the impulse response of the room [17, 18]. Dereverberation can be single channel or multi-channel. The most efficient way of compensating for environmental mismatch due to acoustic reverberation is to train acoustic models using multi-condition/multi-style training data.

For the REVERB challenge we use multi-condition training data for mismatch compensation due to different room impulse responses (RIR) and different channel conditions. We develop seven recognition systems using the Kaldi toolkit based on the following seven front-ends: conventional mel-filterbank (MFB) with log compression, multitaper spectrum estimator-based mel-filterbank (MMFB) with logarithmic nonlinearity ($MMFB_l$), MMFB with power-law nonlinearity ($MMFB_p$), robust compressive gammachirp filterbank (RCGFB), robust mel filterbank (RMFB), iterative deconvolution-based dereverberated MFB (ITD-MFB), and maximum likelihood inverse filtering-based dereverberated (MLIFD) cepstral coefficients. We combine the results of all seven systems to get lowest possible word error rates (WER).

In [2], it was shown that an i-vector characterizing a speaker can be used as an additional input to the feature layer of the DNN (deep neural nets) in order to adapt the DNN to the speaker. This adaptation was found to be effective and helped to boost the performance by approximately 2.0%. In this work we also incorporate this adaptation method.

For the 2-channel and 8-channel we do not apply any multi-microphone signal processing algorithms for doing the dereverberation rather we combine the results from all 2 and all 8 channels, respectively, to get the best possible results. This is found effective and helps to reduce the WER by 1~3%.

2. FRONT-ENDS

2.1. Multitaper Filterbank Features

A windowed direct spectrum estimator is the most often used power spectrum estimation method in speech processing applications. The windowed periodogram estimate can be expressed as:

$$\hat{S}_d(f) = \left| \sum_{j=0}^{N-1} w(j)\, s(j)\, e^{-i\frac{2\pi jf}{N}} \right|^2, \qquad (1)$$

where $f \in \{0, 1, \cdots, K-1\}$ denotes the discrete frequency index, N is the frame length, s(m,j) is the time domain speech signal and $w(j)$ denotes the time domain window function, also known as the taper. The taper, such as the *Hamming* window, is usually symmetric and decreases towards the frame boundaries.

Windowing reduces the bias (the bias of an spectrum estimator $\hat{\theta}$ is defined as the expected difference between the estimated value and the true value of the spectrum θ being estimated and is defined as $\text{bias}\left(\hat{\theta}\right) = E\left[\left(\hat{\theta} - \theta\right)\right]$), but it does not reduce the variance of the spectral estimate [11-12] and therefore, the variance of the cepstral/filterbank features computed from this estimated spectrum remains large. One way to reduce the variance is to replace the windowed periodogram estimate by a so-called multi-taper spectrum estimate [11-12]. It is given by

$$\hat{S}_{MT}(f) = \sum_{p=1}^{M} \lambda(p) \left| \sum_{j=0}^{N-1} w_p(j)\, s(j)\, e^{-\frac{i2\pi jf}{N}} \right|^2, \qquad (2)$$

where N is the frame length and w_p is the pth data taper ($p = 1, 2, ..., M$) used for the spectral estimate $\hat{S}_{MT}(\cdot)$, also known as the pth eigenspectrum. Here, M denotes the number of tapers and $\lambda(p)$ is the weight of the pth taper. The tapers $w_p(j)$ are typically chosen to be orthonormal so that, for all p and q,

$$\sum_j w_p(j)\, w_q(j) = \delta_{pq} = \begin{cases} 1, & p = q \\ 0, & \text{otherwise.} \end{cases}$$

The multitaper spectrum estimate is therefore obtained as the weighted average of M individual spectra. A multi-taper spectrum estimator is somewhat similar to averaging the spectra from a variety of conventional tapers such as Hamming and Hann tapers, but in this case, there will be strong redundancy as the different tapers are highly correlated (they have a common time-domain shape). Unlike conventional tapers, the M orthonormal tapers used in a multi-taper spectrum estimator provide M statistically independent (hence uncorrelated) estimates of the underlying spectrum. The weighted average of the M individual spectral estimates $\hat{S}_{MT}(f)$ then has smaller variance than the single-taper spectrum estimates $\hat{S}_d(f)$ by a factor that approaches $1/M$, i.e., $\text{var}\left(\hat{S}_{MT}(f)\right) \approx \frac{1}{M}\text{var}\left(\hat{S}_d(f)\right)$ [11]. Multitaper Mel filterbank (MMFB) features are then computed from a multiple windowed (e.g., Thomson) spectrum estimate instead of the Hamming windowed periodogram estimate as used in the conventional Mel filterbank (MFB) / cepstral features. In this work two variants of MMFB are used:

***MMFB$_l$*:** MMFB features with logarithmic nonlinearity

***MMFB$_p$*:** MMFB features with power function nonlinearity

Both of the MMFB features were normalized using short-time mean and scale normalization (STMSN) method [10] with a sliding window of 1.5 seconds duration. Our baseline system uses conventional MFB features extracted using the Kaldi toolkit [4].

2.2. RCGFB and RMFB Features

Robust compressive gammachirp filterbank (RCGFB) and robust mel filterbank (RMFB) features were computed following a similar framework to the robust compressive gammachirp filterbank cepstral coefficients (RCGCC) features proposed in [9]. Fig. 1 presents the block diagram for the RCGFB and RMFB feature extractors that incorporate a sigmoid shape suppression rule based on subband *a posteriori* signal-to-noise ratios (SNRs) in order to enhance the auditory spectrum.

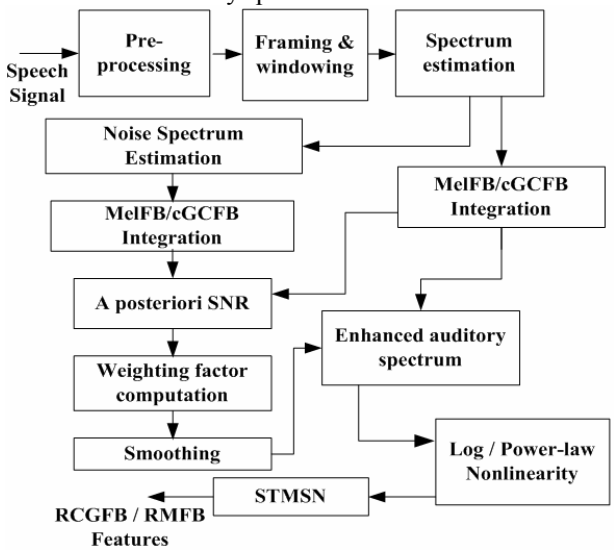

Fig. 1. Robust compressive gammachirp filterbank (RCGFB) and robust mel filterbank (RMFB) features extraction process.

RCGFB utilizes a power function nonlinearity with a coefficient of 0.07 to approximate the loudness nonlinearity of human perception whereas RMFB uses a logarithmic nonlinearity. For feature normalization a short-term mean and scale normalization (STMSN) technique is used with a sliding window of 1.5 seconds.

Under mismatched conditions, this helps to remove the difference of log spectrum between the training and test

environments by adjusting the short-term mean and scale [10].

2.3. Iterative Deconvolution (ITD)-based features

The iterative deconvolution (ITD)-based dereverberated Mel filterbank features extraction method is presented in figure 2. It was proposed in [19]. A Gammatone filterbank integrated auditory spectrum is computed for each windowed frame. ITD is then applied to each subband in the gammatone frequency domain. ITD, an iterative least square approach that minimizes the errors [19]:

$$e_k = \sum_i \left(S(i,k) - \sum_m X(m,k) H(i-m,k) \right)^2, \qquad (3)$$

initialized by nonnegative matrix factorization (NMF), is used to estimate the clean signal $X(m,k)$ and room impulse response $H(m,k)$ from the reverberated signal $S(m,k)$ where k is the subband index and m is the frame index. After reconstructing the dereverberated signal 23-dimensional mel filterbank features are then computed using the Kaldi toolkit.

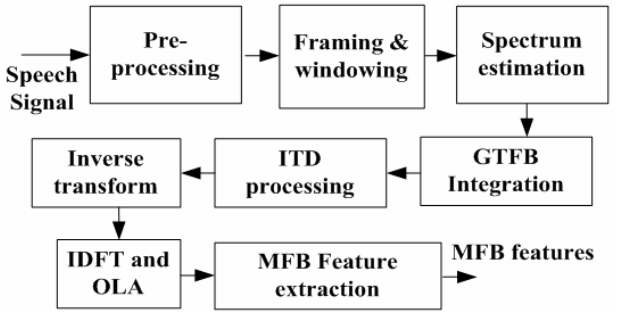

Fig. 2. Iterative Deconvolution (ITD)-based dereverberated MFB (ITD-MFB) features extraction.

2.4. Maximum likelihood inverse filtering-based dereverberated (MLIFD) features

Maximum likelihood inverse filtering-based dereverberation of a reverberated signal in the cepstral domain, proposed in [18], is shown in fig. 3. The purpose of cepstral post-filtering is to partially decorrelate the features. If P $\left(P(z) = 1/\left(1 + pz^{-1}\right)\right)$ is an IIR (inverse impulse response) dereverberation filter of M taps long then the dereverberated cepstral features $c^d[m]$ can be given as:

$$c^d[m] = c[m] - \sum_{k=1}^{M-1} p[k] c^d[m-k], \qquad (4)$$

where m is the frame index, $c[m]$ is the cepstral features of m-th frame of a reverberated speech signal.

Parameters that best describe P can be obtained by finding the most likely Gaussian Mixture Models (GMM) state for all the frames [18].

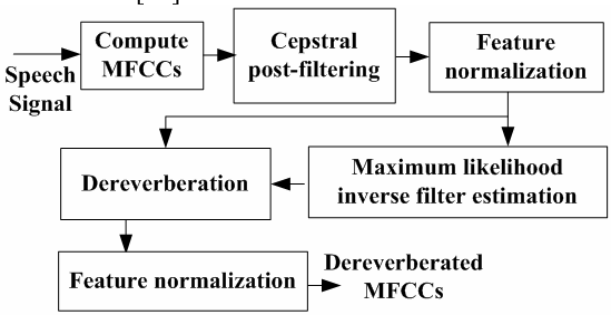

Fig. 3. Maximum likelihood inverse filtering-based dereverberated (MLIFD) MFCCs features extraction.

3. EXTRACTION OF I-VECTORS

The idea of i-vector extraction is based on a simplified version of Joint Factor Analysis (JFA) [3, 6]. Contrary to JFA, different sessions of the same speaker are considered to be produced by different speakers. Rather than making a distinction between the speaker and channel effects the total variability space in the i-vector extraction method simultaneously captures the speaker and channel variabilities [3, 6]. Given a C component GMM-UBM model λ with $\lambda_c = \{w_c, \mu_c, \Sigma_c\}$, $c = 1, 2, ..., C$ and an utterance having a sequence of T feature frames $\{y_1, y_2, ..., y_T\}$ the zeroth and centered first order Baum-Welch statistics on the UBM are computed as:

$$N_c = \sum_{k=1}^{T} p(c \mid y_k, \lambda)$$

$$F_c = \sum_{k=1}^{T} p(c \mid y_k, \lambda)(y_k - \mu_c). \qquad (5)$$

The generative model for the i-vector can be expressed as:

$$M = \mu_c + \mathbf{T}\theta, \qquad (6)$$

where M is a supervector constructed by appending together the first order statistics for each mixture component c, the columns of the low rank total variability matrix **T** span the subspace where most of the speaker specific information lives (along with channel effects). For each speech recording r, an i-vector i_r is obtained as the MAP estimate of θ. Fig. 4 presents a block diagram showing various steps of the i-vector extraction process from the MFCC features. The Features used for i-vector extraction are the 60-dimensional Multitaper MFCCs (MMFCCs) including the 0th cepstral coefficients, delta and double delta features. 512-component diagonal UBM was trained on all the training data. For training the i-vector extractor (i.e., the total variability matrix **T)** we have used all the multi-condition training data generated using the scripts provided

by the REVERB Challenge 2014 organizer. For this task we have generated both speaker-specific and utterance-specific i-vectors

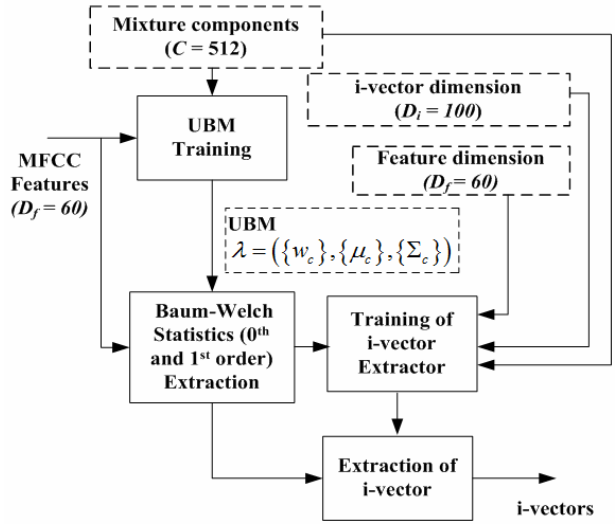

Fig. 4. An i-vector extractor.

4. TRAINING AND DECODING ALGORITHMS

To get best possible results we ran multiple recognition experiments with different features. We then combined the decoded transcripts using ROVER, a recognition system combination software available from NIST. All the training and decoding was done using the Kaldi toolkit. Every recognizer used DNN-HMM hybrid architecture. For all the DNN's that use TRAP (TempoRAl Pattern) features [1] computed from filterbank features, we also input a 100-dimensional i-vector [2] derived from all the utterances of the speaker. This i-vector characterizing a speaker [3, 22] helps the DNN to adapt to the speaker characteristics.

For decoding, we used a pruned trigram LM with 709K trigrams generated from Wall Street Journal language model (LM) training data. The resulting lattices were rescored using a larger trigram LM (with 3.15 million trigrams) generated from the Wall Street LM training data. We used a vocabulary size of 20K words.

4.1. Training data

For training the DNN models, we use the multi-condition training data generated from the clean training data using the scripts provided by the organizer [13, 14]. The Total number of recordings in training data is 70155; 594 utterances are held out in order to provide a cross validation set for DNN training. The same training and validation sets are used for training the DNN for different feature parameters derived from the raw signal. This training data corresponds to training with 20K vocabulary recognition task [13-14].

4.2. Training with MFCC features

The maximum likelihood inverse filtering-based dereverberated MFCC features (i.e., MLIFD features) are used to train a DNN-HMM hybrid system. In order to train the DNN-HMM hybrid system, we first train a GMM-HMM system with 3435 tied triphone states and 40K Gaussians. The training process uses the Kaldi toolkit [4] and the training process is similar to that outlined in [5]. In short, the GMM-HMM is trained on features obtained by first normalizing MFCCs to zero mean per speaker, then splicing together 7 frames of 13-dimensional MFCCs and reducing them to 40 dimensions through LDA, followed by a semi-tied covariance (STC) transform. The resulting features are then transformed through an FMLLR transform. The LDA+STC+FMLLR transformed MFCC features are then used to train the GMM-HMM with 3435 tied triphone states and 40K Gaussians. The deep neural net (DNN) is also trained on the LDA+STC+FMLLR transformed MFCC features. These features are globally normalized to have zero mean and unit variance. The input to the neural net is 11 frames (or 440 feature values). The DNN is initialized with stacked RBMs. The resulting DNN (with 5 hidden layers, 1024 neurons in each hidden layer, 3435 outputs in the output layer) goes through one iteration of sequence training with MPE criteria. The training data is re-aligned with the new DNN and we go through two more iterations of sequence training with the MPE criteria. The resulting DNN-HMM hybrid system is then used for recognition.

4.3. Training with filterbank features

For training DNN-HMM models from the baseline (i.e., conventional mel filterbank (MFB)) features, from the $MMFB_l$ (multi-taper Mel-filterbank with logarithmic nonlinearity) and $MMFB_p$ (multi-taper Mel-filterbank with power-law nonlinearity), from the RCGFB (robust compressive gammachirp filterbank) and RMFB (robust Mel-filterbank) features, and from the ITD-based dereverberated MFB (ITD-MFB) features, we generate 23-dimensional filterbank features per frame for each of the above mentioned front-ends. The process for generating the DNN from these filterbank features is similar to that outlined in [2] for DNN-HMM system with speaker adaptation. From the filterbank features, we compute the TRAP features [1] as follows: we first normalize the 23-dimensional filterbank features to zero mean per speaker. Then 31 frames of these 23-dimensional filterbank features (15 frames on each side of current frame) are spliced together to form a 713-dimensional feature vector. This 713-dimensional feature vector is transformed using a hamming window (to emphasize the center), passed through a discrete cosine transform and the dimensionality is reduced to 368. This 368-dimensional feature vector is globally normalized to have zero mean and unit variance. This normalized 368-dimensional feature vector together

with a 100 dimensional i-vector [6] (computed from the full utterance) is then input to the 7-layer DNN as shown in Fig. 5. The i-vector was length normalized by dividing the i-vector by the square root of the sum of the squares of its elements. Note that the un-normalized i-vectors are approximately Gaussianized by length normalization [7]. The DNN has 5 hidden layers with 1024 neurons each and the output softmax layer has around 3500 outputs. The i-vector characterizing a speaker is used as an additional input to the feature layer in order to adapt the DNN to the speaker. The resulting DNN was then used to re-align the training data. This was then followed with four iterations of sequence training with the MMI criteria.

We generated DNNs with two different variants. In the first variant, the i-vectors were computed from only the current utterance. In the second variant, all the utterances in one room per test condition corresponding to the same speaker were used to compute the 100-dimensional length normalized i-vector for this speaker.

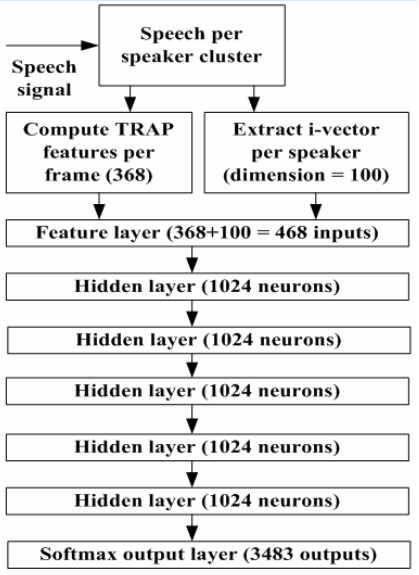

Fig. 5. Architecture of 7-layer DNN used with TRAP and i-vector features.

4.3. Decoding Algorithm

The decoding algorithm is slightly different depending on whether we are doing utterance-based batch processing or full batch processing. For the MLIFD (maximum likelihood inverse filtering-based dereverberation) features that use FMLLR transform, we only use full batch processing, since we need to compute the FMLLR transform for the speaker from all the utterances of the speaker in the room. Computing FMLLR transform from a single short utterance gives poor results (we need over 20 secs of audio to estimate reasonable FMLLR transforms).

For full batch processing, we first diarize all the utterances in a room per test condition using a modified version of the multi-stage segmentation and clustering system [8]. The modification is that each utterance corresponds to one speaker. There is no sub-segmentation of the utterance. Fiterbank features in each speaker cluster are then normalized to zero. We also compute one 100-dimensional length normalized i-vector per speaker [6].

For utterance based batch processing, each utterance was labeled as a different speaker, and an i-vector was computed per utterance. In this case, the filterbank features are normalized per utterance and the 100 dimensional length normalized i-vector is computed per utterance.

5. EXPERIMENTS AND RESULTS

Front-ends or feature extractors used for this task have already been presented in section 2. Kaldi recognizer was used for training and recognition task. Experiments were carried out on the REVERB challenge 2014 corpora [13-16] and results were reported on the all evaluation conditions for the 1-channel, 2-channel, and 8-channel tasks. Results reported here are only on the Evaluation corpus. Development corpus was used for tuning the system parameters.

5.1. Results with utterance-based batch processing

For utterance based batch processing, we decoded each utterance using six different feature sets (baseline, $MMFB_l$, $MMFB_p$, RMFB, RCGFB, and ITD-MFB then combined the six different results using ROVER (Recognizer Output Voting Error Reduction). We did not use MLIFD features as they provided poor results. In ROVER, we ignored the timing information and just used the voting mechanism. So for single channel results, we combine 6 different results using ROVER. Table 1 (a) shows the results for each feature and each room. The last row shows the results after combining results from all 6 features using ROVER. After ROVER, the average *SimData* WER is 10.0% and average *RealData* WER is 27.1%.

For the 2-channel utterance-based recognition, the only difference from 1-channel processing is that for each room we combine the results for the two channels using ROVER. Here, the individual feature parameter WER is not going to be much different from that for 1-channel case, since ROVER needs at least 3 inputs to reduce the WER significantly. However, when we combine all the feature parameter outputs with ROVER (last row), the combination is for 12 different recognizers (2 channels x 6 features). The order of combination in ROVER is the order of the rows in Table 1 (b). As we can see from this table, The average WER for SimData has reduced to 9.6%, and for *RealData* the WER has reduced to 25.6%.

In the 8-channel utterance based recognition, for each feature parameter, we combine the results from all 8 channels using ROVER. The ordering of this combination is from channel 1 to channel 8. We did not try varying this order. Each row shows this combined result. Combining 8-

channels with ROVER reduces the WER significantly. For combining recognition results from all the features (last row), there are 48 recognition outputs (8 channels x 6 features) to combine. The recognition outputs are combined in the same order as the rows in Table 1 (c). That is, channels 1 through 8 of RCGFB are combined first, followed by channels 1-8 of $MMFB_l$ and so on. ROVER is somewhat sensitive to the order of combination. So we combine the best systems first. As we can see from the table, The average WER for *SimData* has reduced to 9.1%, and for *RealData* the WER has reduced to 24.0%.

5.2. Results with full batch processing

There are a few differences *between utterance-based batch processing and full* batch processing. In utterance-based batch processing, we normalize the features of each utterance to zero mean, and compute a 100-dimensional i-vector from this utterance. In full batch processing, we normalize the features of each speaker in a room to zero mean, and compute a 100-dimensional i-vector from this speaker in the room. In order to assign utterances in a room to speakers, we carry out speaker diarization using a modified version of the multi-stage segmentation and clustering system [8] as described before.

In utterance based batch processing, we computed i-vectors separately for each utterance from three different features, and the corresponding i-vector was used when recognizing using that feature. In full batch processing, we computed i-vector for each speaker using the multitaper MFCCs features, and used these i-vectors during training/recognition using other features. This strategy did not work well. Only the WER for $MMFB_l$ features went down while the results for other features were worse than utterance-based processing. Therefore, we used the results from utterance based batch processing for the other features (note MLIFD does not use i-vectors).

Another difference between utterance-based versus full batch processing is that we are able to decode with MLIFD features in full batch processing. The MLIFD features for an utterance are transformed using LDA+STC+FMLLR before input to the neural net. FMLLR transform per utterance resulted in significant increase in WER and therefore MLIFD feature was not used in utterance-based batch processing. In full batch processing, the FMLLR is computed from all the utterances of a speaker in the room. In this scenario, MLIFD features gave very good results. The single channel results are shown in Table 2 (a). In the last column of Table 2 (a), we are combining results from seven different features using ROVER in the same order as the rows in this table. Overall, full batch processing reduced WER from 10.0% to 9.3% as compared to utterance based batch processing.

In the 2-channel full batch processing, the only difference from 1-channel processing is that for each room we combine the results for the two channels using ROVER. Here, the individual feature parameter WER is not going to be much different from that for 1-channel case, since ROVER needs at least 3 inputs to reduce the WER significantly. However, when we combine all the feature parameter outputs with ROVER, the combination is for 14 different recognizers (2 channels x 7 features). The results are shown in the last row of Table 2 (b). As we can see from the table, The average WER for SimData has reduced from 9.6% to 9.0%, and for *RealData* WER reduced from 25.6% to 23.4% when compared with 2-channel utterance based processing.

For the 8-channel full batch processing, for each feature parameter, we combine the results from all 8 channels using ROVER. The ordering of this combination is from channel 1 to channel 8. We did not try varying this order. Each row in Table 2 (c) presents this combined result. Combining 8-channels with ROVER reduces the WER significantly. For combining with ROVER recognition results from all the features (last row in Table 2 (c)), there are 56 recognition outputs (8 channels x 7 features) to combine. However, for some reason, we can only combine a maximum of 50 recognition outputs in ROVER. The recognition outputs are combined in the same sequence as the rows in Table 2 (c). That is, channels 1 through 8 of MLIFD are combined first, followed by channels 1-8 of RCGFB and so on. So only the first two channels of the Baseline features are combined with ROVER. As we can see, compared to 2 channel full batch processing, the WER for *SimData* has reduced from 9.0% to 8.9%, and for *RealData* WER has reduced from 23.4% to 21.7%.

6. CONCLUSION

In this work, to get the best possible recognition results (i.e., lowest WER), we decided to use multiple front-ends -based recognition systems and then combined the recognition results using ROVER. Front-ends chosen for the REVERB challenge task were: convention MFB, MMFB (with log and power-law nonlinearity), RCGFB, RMFB, ITD-MFB, and MLIFD features. We also applied i-vector -based speaker adaptation as it was found to boost the performance by approximately 2% [2]. In the case of 2- and 8- channel tasks, to get benefited from more than one channel data, we also exploited ROVER instead of any multi-microphone signal processing method. Compared to the baseline all other front-ends performed better in terms of WER both in SimData (except MMFB with power function nonlinearity) and RealData. The best results were obtained with the full batch processing and with 8-channel task. For 8-channel task we obtained an average WER of 8.9% and 21.7% on the SimData and RealData, respectively.

Our future work is to try some multi-channel dereverberation algorithms and see their the recognition

performances both in clean- and multi-condition training mode.

7. REFERENCES

[1] F. Grezl, "TRAP-based Probabilistic Features for Automatic Speech Recognition", Doctoral Thesis, dept. Computer Graphics & Multimedia, Brno Univ of Technology, Brno 2007.

[2] V. Gupta, P.Kenny, P. Ouellet, T. Stafylakis, "I-Vector-based speaker adaptation of deep neural networks for French broadcast audio transcription", Proc. ICASSP (to appear), 2014.

[3] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," IEEE Trans. Audio, Speech, Lang. Process., vol. 19, no. 4, pp. 788–798, May 2011.

[4] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hanneman, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer and K. Vesely, "The Kaldi Speech Recognition Toolkit", Proc. ASRU, 2011.

[5] K. Vesel´y, A. Ghosal, L. Burget, D. Povey, "Sequence discriminative training of deep neural networks", Proc. Interspeech 2013, pp. 2345–2349.

[6] P. Kenny, "A small footprint i-vector extractor", Proc. Odyssey 2012, Singapore.

[7] D. Garcia-Romero, and C. Y. Espy-Wilson, "Analysis of ivector length normalization in speaker recognition systems", Proc. Interspeech-2011.

[8] V. Gupta, G. Boulianne, P. Kenny, P. Ouellet, P. Dumouchel, "Speaker Diarization of French Broadcast News", Proc. ICASSP, pp. 4365–4368, 2008.

[9] M. J. Alam, P. Kenny, D. O'Shaughnessy, "Robust Feature Extraction for Speech Recognition by Enhancing Auditory Spectrum," Proc. INTERSPEECH, September 2012.

[10] Alam, J., Ouellet, P., Kenny, P., O'Shaughnessy, D., "Comparative Evaluation of Feature Normalization Techniques for Speaker Verification," *Proc NOLISP*, LNAI 7015, pp. 246-253, Las Palmas, Spain, November 2011.

[11] Md. J. Alam, T. Kinnunen, P. Kenny, P. Ouellet, D. O'Shaughnessy, "Multitaper MFCC and PLP Features for Speaker Verification Using i-Vectors", *Speech Communication*, 55(2): 237--251, February 2013.

[12] Md. J. Alam, Kenny, P., and O'Shaughnessy, D., "Low-variance Multitaper Mel-Frequency Cepstral Coefficient Features for Speech and Speaker Recognition Systems," Springer Cognitive Computation Journal, December 2012.

[13] Kinoshita, K.; Delcroix, M.; Yoshioka, T.; Nakatani, T.; Habets, E.; Haeb-Umbach, R.; Leutnant, V.; Sehr, A.; Kellermann, W.; Maas, R.; Gannot, S.; Raj, B., "The REVERB Challenge: A Common Evaluation Framework for Dereverberation and Recognition of Reverberant Speech," Proceedings of the WASPAA, 2013.

[14] Robinson, T.; Fransen, J.; Pye, D.; Foote, J.; Renals, S., "WSJCAMO: a British English speech corpus for large vocabulary continuous speech recognition," Proc. ICASSP, pp. 81-84, 1995.

[15] Lincoln, M.; McCowan, I.; Vepa, J.; Maganti, H.K., "The multi-channel Wall Street Journal audio visual corpus (MC-WSJ-AV): specification and initial experiments," Proc. ASRU, pp. 357-362, 2005.

[16] Paul, Douglas B.; Baker, Janet M., "The design for the Wall Street Journal-based CSR corpus," Proc. HLT, pp. 357-362, 1992.

[17] P. A. Naylor, N. D. Gaubitch, Speech Dereverberation, Signals and Communication Technology series, Springer; 2010 edition, July 28, 2010.

[18] K. Kumar and R. M. Stern, "Maximum-likelihood-based cepstral inverse filtering for blind speech dereverberation," Proc. ICASSP, March 2010, Dallas, Texas.

[19] K. Kumar, B. Raj, R. Singh, and R. M. Stern, An iterative least-squares technique for dereverberation, Proc. ICASSP, May 2011, Prague, Czech Republic.

[20] K. Kumar, and R. M. Stern, "Environment-invariant compensation for reverberation using linear post-filtering for minimum distortion," Proc. ICASSP, April 2008, Las Vegas, Nevada.

[21] M. J. Alam, P. Kenny, V. Gupta, P. Dumouchel, D. O'Shaughnessy, "Comparative Evaluation of Several Robust Feature Extractor for Continuous Speech Recognition," submitted to ICASSP, 2014.

[22] G. Saon, H. Soltau, D. Nahamoo and M. Picheny, "Speaker Adaptation of Neural Network Acoustic Models using I-vectors," Proc. ASRU, 2013.

Table 1 *WER obtained using utterance-based batch processing for (a) 1channel (b) 2 channel, and (c) 8 channel tasks.*

(a)	**SimData**							**RealData**		
	Room 1		Room 2		Room 3		Avg.	Room 1		Avg.
	Near	Far	Near	Far	Near	Far		Near	Far	
RCGFB	8.2	9.4	9.8	15	10.8	16.6	11.6	30.4	31.5	30.9
$MMFB_l$	8.3	9.3	9.9	15.7	10.6	17.6	11.9	31.6	30.9	31.2
$MMFB_p$	8.5	9.8	11.1	17.2	11.8	19.2	12.9	30.2	31.9	31
RMFB	8.4	9.1	9.7	15	10.8	17	11.6	31.8	31.3	31.5
ITD-MFB	7.6	8.8	10.4	14.5	9.8	16	11.1	31.9	33	32.4
Baseline	7.6	8.9	11.5	18.1	11.2	18.8	12.6	41	38	39.4
ROVER-all	**7.1**	**8.1**	**8.9**	**12.9**	**9.2**	**13.8**	**10**	**27.2**	**26.9**	**27.1**

(b)	**SimData**							**RealData**		
	Room 1		Room 2		Room 3		Avg.	Room 1		Avg.
	Near	Far	Near	Far	Near	Far		Near	Far	
RCGFB	8.4	9.5	10.1	15.2	11.1	17.1	11.9	31.4	32.4	31.9
$MMFB_l$	8.5	9.3	10.1	15.9	11.3	17.9	12.2	32.8	31.2	32

$MMFB_p$	8.9	10	11	18.1	12.5	20.3	13.5	31.8	32.9	32.3
RMFB	8.4	9.1	10	15.4	10.8	17.3	11.9	32.6	31	31.8
ITD-MFB	7.8	9	10.5	15.1	10.4	16.1	11.5	33	32.6	32.8
Baseline	7.8	9.2	11.6	18.3	11.7	19.3	13	42.4	38.5	40.4
ROVER-all	**7**	**7.8**	**8.4**	**12.1**	**9**	**13.2**	**9.6**	**25.5**	**25.7**	**25.6**

(c)	**SimData**							**RealData**		
	Room 1		Room 2		Room 3		Avg.	Room 1		Avg.
	Near	Far	Near	Far	Near	Far		Near	Far	
RCGFB	8.1	9	9.1	14	10.3	15.3	11	27.9	28.7	28.3
$MMFB_l$	8.1	8.7	9.3	14.3	10.1	15.8	11.1	28.4	27	27.7
$MMFB_p$	8.4	9.2	10.2	16.1	11.4	18	12.2	27.5	28.7	28.1
RMFB	8.1	8.7	9.1	13.6	10	14.8	10.7	29.4	27.7	28.5
ITD-MFB	7.2	8.1	9.7	13.1	9.5	14.8	10.4	29.8	30.1	30
Baseline	7.6	8.4	10.6	17	10.6	17.9	12	37.8	36.8	37.3
ROVER-all	**6.7**	**7.3**	**8.3**	**11.6**	**8.6**	**12.6**	**9.1**	**23.8**	**24.1**	**24**

Table 2 *WER obtained using full batch processing for (a) 1channel (b) 2 channel, and (c) 8 channel tasks.*

(a)	**SimData**							**RealData**		
	Room 1		Room 2		Room 3		Avg.	Room 1		Avg.
	Near	Far	Near	Far	Near	Far		Near	Far	
MLIFD	8	8.8	10.9	15.9	11.1	17.6	12	27	28	27.5
RCGFB	8.2	9.4	9.8	15	10.8	16.6	11.6	30.4	31.5	30.9
$MMFB_l$	8.7	9.9	10.3	17.2	11.3	18.7	12.6	28.7	28.7	28.6
$MMFB_p$	8.5	9.8	11.1	17.2	11.8	19.2	12.9	30.2	31.9	31
RMFB	8.4	9.1	9.7	15	10.8	17	11.6	31.8	31.3	31.5
ITD-MFB	7.6	8.8	10.4	14.5	9.8	16	11.1	31.9	33	32.4
Baseline	7.6	8.9	11.5	18.1	11.2	18.8	12.6	41	38	39.5
ROVER-all	**6.7**	**7.3**	**8.4**	**11.8**	**8.7**	**12.7**	**9.3**	**23.8**	**24.8**	**24.3**

(b)	**SimData**							**RealData**		
	Room 1		Room 2		Room 3		Avg.	Room 1		Avg.
	Near	Far	Near	Far	Near	Far		Near	Far	
MLIFD	8.2	9	11.1	16.5	11.5	18.4	12.5	27.3	27.5	27.4
RCGFB	8.4	9.5	10.1	15.2	11.1	17.1	11.9	31.4	32.4	31.9
MMFCC	8.8	10.4	10.5	17.9	11.8	19.5	13.2	29.5	28.8	29.1
MMFCC_I	8.9	10	11	18.1	12.5	20.3	13.5	31.8	32.9	32.3
RMFB	8.4	9.1	10	15.4	10.8	17.3	11.9	32.6	31	31.8
ITD-MFB	7.8	9	10.5	15.1	10.4	16.1	11.5	33	32.6	31
Baseline	7.8	9.2	11.6	18.3	11.7	19.3	13	42.4	33	40.4
ROVER-all	**6.6**	**7.4**	**8.1**	**11.2**	**8.5**	**12.2**	**9**	**22.6**	**24.2**	**23.4**

(c)	**SimData**							**RealData**		
	Room 1		Room 2		Room 3		Avg.	Room 1		Avg.
	Near	Far	Near	Far	Near	Far		Near	Far	
MLIFD	7.5	8.3	10	14.1	10.4	15.9	11	23.8	24.4	24.1
RCGFB	8.1	9	9.1	14	10.3	15.3	11	27.9	28.7	28.3
$MMFB_l$	8.5	9.5	9.5	16.1	10.8	17.4	12	26	26.2	26.1
$MMFB_p$	8.4	9.2	10.2	16.1	11.4	18	12.2	27.5	28.7	28.1
RMFB	8.1	8.7	9.1	13.6	10	14.8	10.7	29.4	27.7	28.5
ITD-MFB	7.2	8.1	9.7	13.1	9.5	14.8	10.4	29.8	30.1	30
Baseline	7.6	8.4	10.6	17	10.6	17.9	12	37.8	36.8	37.3
ROVER-all	**6.7**	**7.3**	**8**	**11.1**	**8.1**	**12.1**	**8.9**	**21.4**	**22**	**21.7**