Alam et al. EURASIP Journal on Advances in Signal Processing (2015) 2015:50
DOI 10.1186/s13634-015-0238-6

EURASIP Journal on Advances in Signal Processing
a SpringerOpen Journal

RESEARCH Open Access

Speech recognition in reverberant and noisy environments employing multiple feature extractors and i-vector speaker adaptation

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

Abstract

The REVERB challenge provides a common framework for the evaluation of feature extraction techniques in the presence of both reverberation and additive background noise. State-of-the-art speech recognition systems perform well in controlled environments, but their performance degrades in realistic acoustical conditions, especially in real as well as simulated reverberant environments. In this contribution, we utilize multiple feature extractors including the conventional mel-filterbank, multi-taper 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 coefficient features for speech recognition with multi-condition training data. In order to improve speech recognition performance, we combine their results using ROVER (Recognizer Output Voting Error Reduction). For two- and eight-channel tasks, to get benefited from the multi-channel data, we also use ROVER, instead of the multi-microphone signal processing method, to reduce word error rate by selecting the best scoring word at each channel. As in a previous work, we also apply i-vector-based speaker adaptation which was found effective. In speech recognition task, speaker adaptation tries to reduce mismatch between the training and test speakers. Speech recognition experiments are conducted on the REVERB challenge 2014 corpora using the Kaldi recognizer. In our experiments, we use both utterance-based batch processing and full batch processing. In the single-channel task, full batch processing reduced word error rate (WER) from 10.0 to 9.3 % on SimData as compared to utterance-based batch processing. Using full batch processing, we obtained an average WER of 9.0 and 23.4 % on the SimData and RealData, respectively, for the two-channel task, whereas for the eight-channel task on the SimData and RealData, the average WERs found were 8.9 and 21.7 %, respectively.

Keywords: Speech recognition; Multiple window; Filterbank features; Dereverberation; I-vectors; DNN-HMM; GMM-HMM

1 Introduction

A key component in hands-free man-machine interaction is the automatic speech recognition (ASR). State-of-the-art speech recognition systems are based on statistical acoustic models. These acoustic models are usually 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 the time domain. Compensation of the acoustic reverberant environment is usually done by dereverberation, which can be obtained by inverse filtering the impulse response of the room [1, 2]. 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 [3]. In a multi-style training condition, acoustic models are trained on clean plus noisy (due to reverberation and additive background noise) data and testing is done on all test data. The goal of multi-style training is to create matched training/test environments.

* Correspondence: jahangir.alam@crim.ca
[1]CRIM, Montreal, Quebec, Canada
Full list of author information is available at the end of the article

Although it is expensive to obtain enough representation noisy data that can cover a wide range of noise types, signal-to-noise ratios, and reverberation times, it is an effective method for mismatch compensation [4].

A wide variety of transformations and feature extraction steps has already been employed in Gaussian mixture model-hidden Markov model (GMM-HMM)-based speech recognition systems to extract and normalize the information contained in the input speech signal as efficiently as possible. Recently, the development of deep neural network (DNN)-based effective acoustic model training methods made it possible for the acoustic model to learn relationships between features and phonemes to a higher extent than it is possible with mutually implemented feature transformation steps [5]. The DNNs are less sensitive to the increase in the input dimensionality than GMMs. Therefore, in DNNs, a richer set of features can be exploited than conventional low-dimensional feature vectors, such as mel-frequency cepstral coefficients (MFCCs) and perceptual linear prediction (PLP) coefficients. In particular, in the DNN-HMM framework, the log mel-filterbank (MFB) features have been shown to provide improved recognition accuracy than the MFCC and PLP features in clean as well as in noisy environments [6–8]. Also, The DNN-HMM hybrid architecture with filterbank features has been shown to outperform the GMM-HMM speech recognition framework with MFCC and PLP features [9, 10].

In this work, we use a hybrid DNN-HMM architecture with several variants of filterbank features and one cepstral feature (maximum likelihood inverse filtering-based dereverberated (MLIFD) cepstral coefficients) for the REVERB challenge 2014 tasks.

For the REVERB challenge, we use multi-condition training data for mismatch compensation due to different room impulse responses (RIRs), signal-to-noise ratios, and different channel conditions. We develop seven recognition systems using the Kaldi toolkit based on the following seven features:

- Conventional mel-filterbank (MFB) with log compression,
- Multi-taper 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 the lowest possible word error rates (WERs). For multi-channel tasks, we also exploit ROVER (Recognizer Output Voting Error Reduction), instead of the multi-microphone signal processing methods such as beamforming, to choose the best scoring word (with the highest number of votes) at each channel of the multi-channel task. ROVER seeks to reduce the WERs by exploiting differences in the nature of error made by multiple recognition systems [11]. ROVER has also been exploited in [12] to reduce WERs in the REVERB challenge tasks.

Speaker adaptation in speech recognition task helps to reduce the mismatch between the training and test speakers and results in improved recognition performance for test speakers. In [13], it was shown that an i-vector characterizing a speaker can be used as an additional input to the feature layer of the DNNs 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 two-channel and eight-channel tasks, we do not apply any multi-microphone signal processing algorithms for doing the dereverberation, but rather, we utilize ROVER to combine the results from all two and all eight channels, respectively, to get the best possible results. This is found effective and helps to reduce the WER by 1~3 %.

The rest of the article is organized as follows: We first describe the various cepstral, filterbank feature extraction and single-channel dereverberation techniques considered in this work in Section 2. In Section 3, we provide a brief description on how to train and extract low-dimensional i-vectors from the training and test recordings. The DNN-HMM-based training and decoding algorithms have been presented in Section 4, and in Section 5, the experimental results are reported and discussed. The article is concluded in Section 6.

2 Feature extractors

Extraction of useful information from speech has been a subject of active research for many decades. The first step in an automatic speech recognition system is a feature extractor which transforms a raw signal into a compact representation. The most popular features used in a GMM-HMM recognizer are MFCCs, and the mel-filterbank energy is the widely used feature in a DNN-HMM recognizer. In this section, we describe the features (cepstral as well as filterbank) and single-channel blind dereverberation algorithms used in the REVERB challenge 2014.

2.1 Multi-taper filterbank features

The most often used power spectrum estimation method in speech processing applications is a windowed direct

spectrum estimator, and it can be expressed mathematically as

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

where $f \in \{0, 1, \cdots, K-1\}$ denotes the discrete frequency index, N is the frame length, $s(j)$ 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 a 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 [14, 15]; 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 [14, 15]. It is given by

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

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

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

The multi-taper 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).

Figure 1 depicts the multi-taper spectrum estimation process from a frame of speech signal with $M = 6$ orthogonal tapers. 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 that of the single-taper spectrum estimates $\hat{S}_d(f)$ by a factor that approaches $1/M$, i.e., $\text{var}\left(\hat{S}_{MT}(f)\right) \approx \frac{1}{M}\text{var}\left(\hat{S}_d(f)\right)$ [14]. According to [16], variance in the feature vectors has a direct bearing to the variance of the Gaussian modeling speech classes. In general, reduction in feature vector variance increases class separability and thereby increases recognition accuracy [16]. Multi-taper 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. The motivation behind using the multi-taper method in the REVERB challenge 2015 tasks is its improved speaker recognition performance on the microphone speech portions of the NIST-SRE corpora [14, 15]. In this work, two variants of MMFB are used:

MMFBl: MMFB features with logarithmic nonlinearity

MMFBp: MMFB features with power function nonlinearity

In this work, we use the Thomson multi-taper method. In the Thomson multi-taper method, a set of M orthonormal data tapers with good leakage properties are specified from the *Slepian sequences* [17]. Slepian sequences are defined as the real, unit-energy sequences on $[0, N-1]$ having the greatest energy in a bandwidth B. Slepian taper can be shown to be the solutions to the following eigenvalue problem:

$$\mathbf{A}_{nj}w_j^p = \nu^p w_n^p,$$

where $0 \le n \le N-1$, $0 \le j \le N-1$, $\mathbf{A}_{nj} = \frac{\sin 2\pi W(n-j)}{\pi(n-j)}$ is a real symmetric Toeplitz matrix, $0 < \nu^p \le 1$ is the p-th eigenvalue corresponding to the p-th eigenvector w_n^p known as the Slepian taper, W is the half frequency bandwidth, and $\lambda(p) = \nu^p$, $p = 1, 2, \ldots, M$. Slepian sequences (or DPSS), proposed by D. Slepian in [17], were chosen as tapers in [18] as these tapers are mutually orthonormal and possess desirable spectral concentration properties (i.e., it has the highest concentration of energy in the user-defined frequency interval $(-W, W)$). The first taper in the set of Slepian sequences is designed to produce a direct spectral estimator with minimum broadband bias (bias caused by leakage via the sidelobes). The higher order tapers ensure minimum broadband bias while being orthogonal to all lower order tapers. The tapers and taper weights in this method can be obtained using the following MATLAB function:

$$[w\ \lambda] = \text{dpss}(N, \beta, M),$$

where β is the time half bandwidth product. The optimum number of tapers for a continuous speech recognition task was found to be $M = 6$ [4, 14, 15] and $\beta = 3.0$.

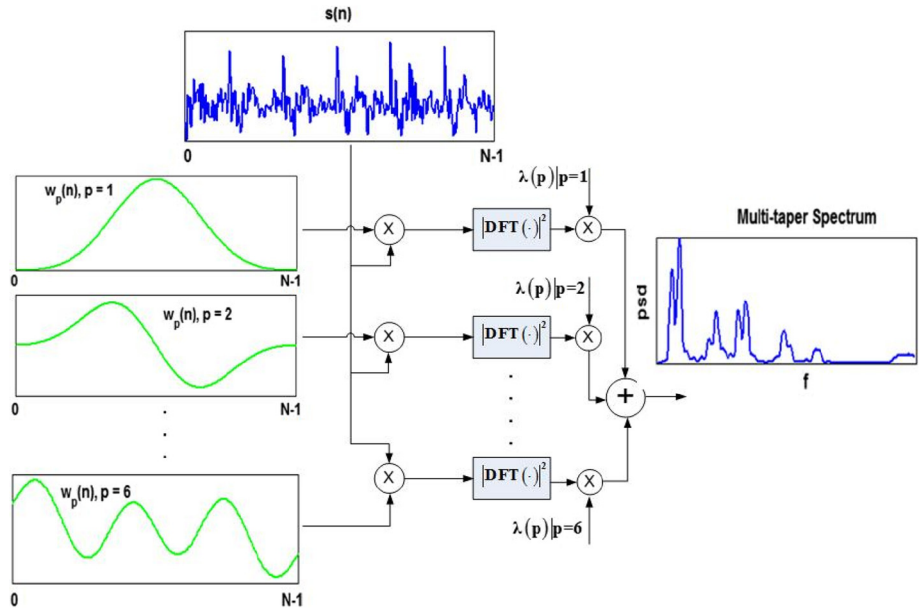

Fig. 1 Block diagram of multi-taper spectrum estimation method [14]

Both of the MMFB features were normalized using the short-time mean and scale normalization (STMSN) method [19] with a sliding window of 1.5 s duration. Our baseline system uses conventional MFB features extracted using the Kaldi toolkit [20]. Various steps for the extraction of MMFB features are shown in Fig. 2. MFB features can be obtained as a special case of MMFB features from Fig. 2 by selecting the number of taper $M = 1$, taper weights $\lambda(p) = 1$, and $w_p(j)$ as the symmetric Hamming taper.

2.2 Robust compressive gammachirp filterbank and mel-filterbank features

Robust compressive gammachirp filterbank (RCGFB) and robust mel-filterbank (RMFB) features were computed following a similar framework to the robust

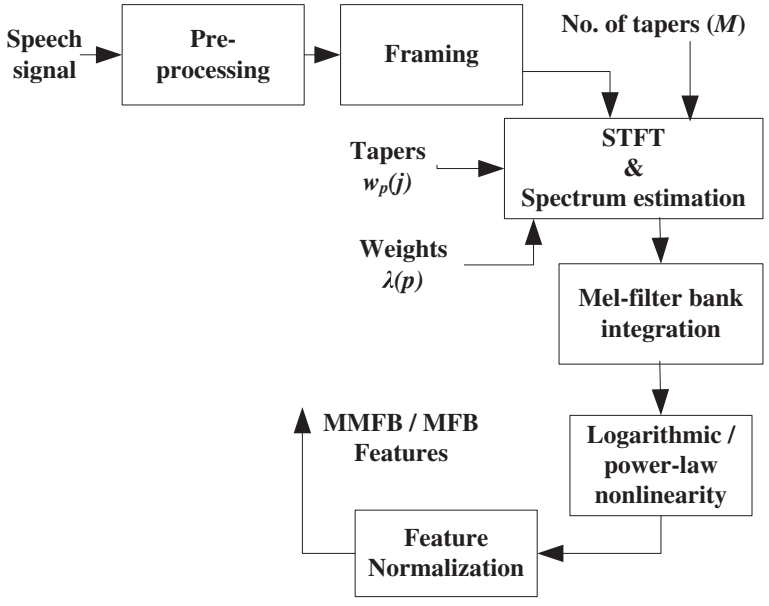

Fig. 2 Block diagram showing various steps for the extraction of multi-taper spectrum estimation-based mel-filterbank (MMFB) features and the conventional mel-filterbank (MFB) features when the number of taper $M = 1$, $\lambda(p) = 1$, and $w_p(j)$ is the Hamming taper

compressive gammachirp filterbank cepstral coefficient (RCGCC) features proposed in [21]. The main motivation for using RCGFB and RMFB feature extractors in the REVERB challenge 2014 tasks is the better recognition accuracy obtained by the RCGCC features on the AURORA-5 and AURORA-4 corpora which represents reverberant acoustic conditions and additive noise as well as different microphone channel conditions, respectively [21, 22].

Figure 3 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.

The sigmoid-shaped weighting rule $H(k, m)$ to enhance the auditory spectrum $S_{as}(k, m)$ based on the subband a posteriori SNR (in dB) $\gamma_{sb}(k, m)$ can be formulated as [2]

$$H(k,m) = \frac{1}{1 + e^{-\frac{\gamma_{sb}(k,m)-5}{\tau}}}, \quad (3)$$

where k is the subband index, m is the frame index, τ is a parameter that controls the lower limit of the weighting function, $\gamma_{sb}(k, m)$ is defined as

$$\gamma_{sb}(k,m) = \max\left(10\log_{10}\left(\frac{S_{as}(k,m)}{N_{as}(k,m)}\right), -4.0\right), \quad (4)$$

and $N_{as}(k, m)$ is the noise power spectrum mapped onto the auditory frequency axis. In order to remove the outliers from the weighting factor $H(k, m)$, as given by Eq. (3), due to noise variability, we use a two-dimensional median filter. For smoothing the decision regions, a two-dimensional moving average filter is also applied [21, 22].

Noise power spectrum estimation from the noisy speech signal plays a very important rule in noise reduction/speech enhancement algorithms. Here, we use a minimum mean square error (MMSE)-soft speech presence probability (SPP) (MMSE-SPP)-based noise estimation approach, proposed in [23], for estimation of noise power spectra. In this method, the initial estimate of the noise power spectrum is computed by averaging the first ten frames of the speech spectrum. The advantage of this method is that it does not require a bias correction term as required by a MMSE-based noise spectrum estimation method; it also results in less overestimation of noise power and is computationally less expensive [24]. The reason behind choosing the MMSE-SPP-based noise spectrum estimation method is that it is computationally simple and requires only one parameter to be tuned.

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 s. Under mismatched conditions, STMSN helps to remove the difference of log spectrum between the training and test environments by adjusting the short-term mean and scale [19].

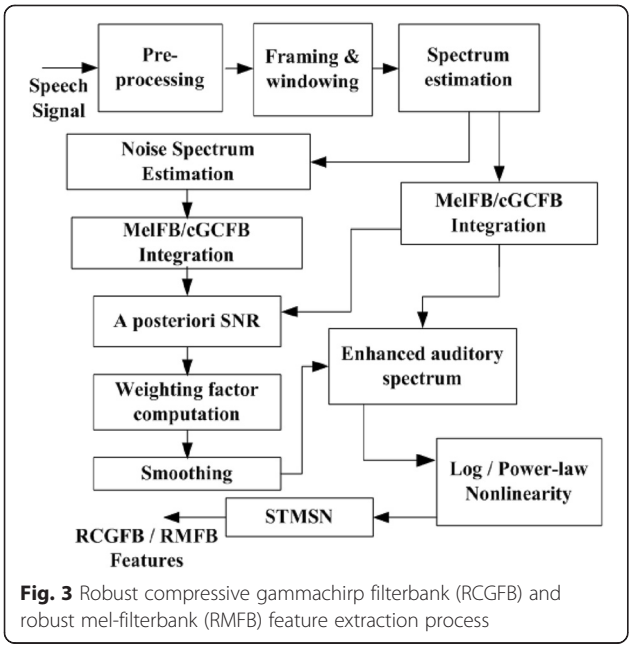

Fig. 3 Robust compressive gammachirp filterbank (RCGFB) and robust mel-filterbank (RMFB) feature extraction process

2.3 Iterative deconvolution-based features

A general nonnegative matrix factorization (NMF) framework decomposes spectra of reverberated speech in to those of the clean and room impulse response filter. An iterative least squares deconvolution technique can be employed for spectral factorization [25]. The iterative deconvolution (ITD)-based dereverberated mel-filterbank feature extraction method is presented in Fig. 4. It was introduced in [25]. A gammatone

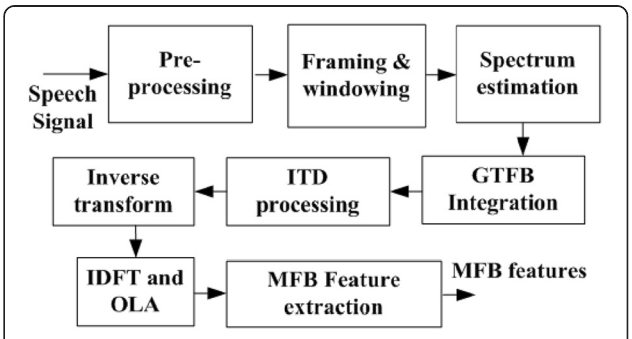

Fig. 4 Iterative deconvolution (ITD)-based dereverberated MFB (ITD-MFB) feature extraction

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 squares approach that minimizes the errors [25]

$$e_k = \sum_i \left(S(i,k) - \sum_m X(m,k) H_r(i-m,k) \right)^2, \quad (5)$$

initialized by NMF, is used to estimate the clean signal $X(m, k)$ and room impulse response $H_r(k, m)$ 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 [4].

2.4 Maximum likelihood inverse filtering-based dereverberated features

Maximum likelihood inverse filtering-based dereverberation of a reverberated signal in the cepstral domain, proposed in [2], is shown in Fig. 5. The purpose of cepstral post-filtering is to partially decorrelate the features. If P ($P(z) = 1/(1 + pz^{-1})$) is an inverse impulse response (IIR) dereverberation filter of M taps long, then the dereverberated cepstral features $c^d[m]$ can be given as

$$c^d[m] = c[m] - \sum_{k=1}^{M-1} p[k] c^d[m-k], \quad (6)$$

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

Parameters that best describe P can be obtained by maximizing the log likelihood with respect to the Gaussian mixture models (GMM) for all the frames of the speech [2]. A common approximation in GMMs is to replace overall GMM likelihood score by the top N scoring Gaussian density among the set of Gaussian mixtures [2]. Here, similar to [2], we use top-1 approximation for filter updates. We apply cepstral mean normalization (CMN) to remove any constant additive shift in the cepstral features and to partially decorrelate the features cepstral post filtering (CPF) [26] is used.

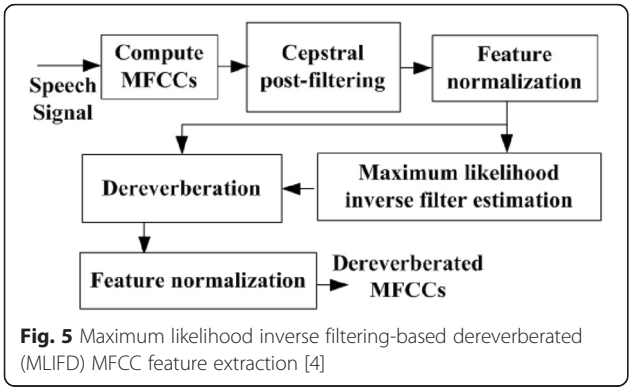

Fig. 5 Maximum likelihood inverse filtering-based dereverberated (MLIFD) MFCC feature extraction [4]

3 Training and extraction of i-vectors

A state-of-the-art speaker verification system is based on the low-dimensional i-vector representation of speech recordings. The i-vector extraction idea is based on a simplified version of joint factor analysis (JFA) [27, 28]. 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 [27, 28]. Given a C component GMM-UBM (universal background models) model λ with $\lambda_c = \{w_c, \mu_c, \Sigma_c\}$, $c = 1, 2,..., C$ (w_c, μ_c, and Σ_c represent the mixture weight, mean vector, and covariance of the c-th Gaussian component, respectively) and an utterance having a sequence of T feature frames $\{y_1, y_2,..., y_T\}$, the zeroth- and centered first-order sufficient statistics (Baum-Welch statistics) on the UBM need to be computed for i-vector modeling:

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

$$F_c = \sum_{k=1}^{T} p(c|y_k, \lambda)(y_k - \mu_c). \quad (7)$$

The speaker and channel-dependent i-vector model can be expressed as

$$M = \mu_c + \mathbf{T}\theta, \quad (8)$$

where M is a supervector constructed by appending together the first-order statistics for each mixture component c, and the columns of the low rank total variability matrix $\mathbf{T}$ span the subspace where most of the speaker-specific information lives (along with channel effects). The posterior of θ is assumed to have standard normal distribution. For each speech recording r, an i-vector i_r is obtained as the MAP estimate of θ and is given by the following equation:

$$i_r = B^{-1} \mathbf{T}^t \Sigma^{-1} F, \quad (9)$$

where $B = I + \mathbf{T}^t \Sigma^{-1} \tilde{N} \mathbf{T}$ is the precision matrix, Σ is a block diagonal covariance matrix with blocks given by the covariance matrices of the UBM, I is the identity matrix, and $\tilde{N}$ and F indicate the zeroth- and first-order sufficient statistics, respectively, of an utterance [27, 29].

The various steps of the i-vector extraction process from the MFCC features are shown in Fig. 6. The features used for i-vector extraction are the 60-dimensional multi-taper MFCCs (MMFCCs) including the zeroth

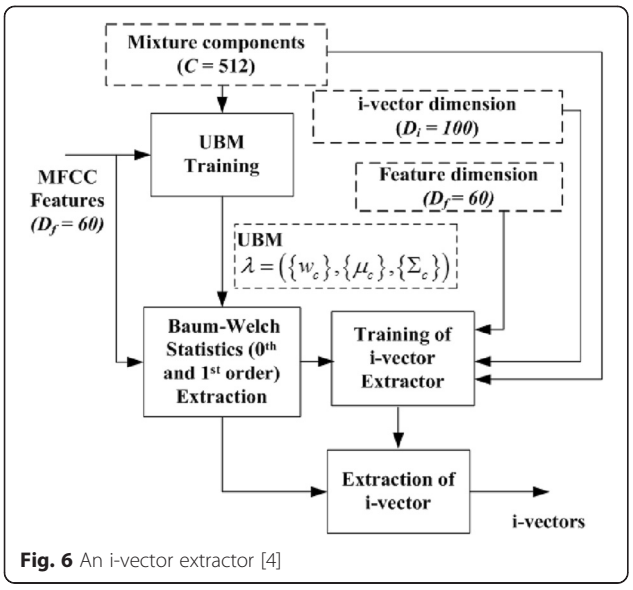

Fig. 6 An i-vector extractor [4]

cepstral coefficients and delta and double delta features. The 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. A speech recording is first diarized, and each speaker cluster is represented by an i-vector. For this task, we have generated both speaker-specific and utterance-specific i-vectors [3, 4]. Acoustic feature vectors are augmented by the corresponding i-vectors before being applied to the DNN.

4 Algorithms used in training and decoding

In order 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 the DNN-HMM hybrid architecture. For all the DNNs that use TRAP (TempoRAl Pattern) features [30] computed from filterbank features, we also input a 100-dimensional i-vector [13] derived from all the utterances of the speaker. This i-vector characterizing a speaker [27, 31] 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 and test data

In REVERB challenge 2014, the task is to recognize read speech in eight different acoustic conditions. Six of which are simulated by convolving the WSJCAM0 corpus [32] with three measured room impulse responses (RIRs) at near- and far-field microphones and by adding stationary noise from the same room at a signal-to-noise ratio (SNR) of 20 dB (SimData). The other two conditions (RealData) correspond to real recordings of speakers in a reverberant meeting room at two microphone distances with ambient, stationary noise taken from the MC-WSJ-AV corpus [33]. The T60 times range from 0.25 to 0.7 s. The T60 times are assumed to be unknown at the test time. In all conditions, eight channels of a circular microphone array are available, of which the first is used as a reference channel. The SimData set has 1484/2176 utterances from 20/20 speakers in the development and evaluation data. The RealData set has 179/372 utterances from 5/10 speakers [34].

For training the DNN models, we use the multi-condition training data generated from the clean training data (WSJCAM0 corpus) using the scripts provided by the organizer [32, 35]. The multi-condition training set contains artificially distorted data similar to the SimData set. It is of the same size as the clean WSJCAM0 training set, containing 7861 utterances from 92 speakers. The RIRs and noise types are chosen randomly with equal probability. The total number of recordings in the training data is 70,155, and 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 tasks [32, 35].

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 [20], and the training process is similar to that outlined in [36]. 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 linear discriminant analysis (LDA) [37], followed by a semi-tied covariance (STC) transform [38]. The resulting features are then transformed through a feature space maximum likelihood linear regression (FMLLR) transform [39]. The LDA + STC + FMLLR [20, 37–40] transformed MFCC features are then used to train the GMM-HMM system having 3435 tied

triphone states and 40K Gaussians. The DNN is also trained on the LDA + STC + FMLLR [20, 37–40] 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, and 3435 outputs in the output layer) goes through one iteration of sequence training with MPE (minimum phone error) criteria. The training data is realigned 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 abovementioned front-ends. The process for generating the DNN from these filterbank features is similar to that outlined in [13] for the DNN-HMM system with speaker adaptation. From the filterbank features, we compute the TRAP features [30] 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 [28] (computed from the full utterance) is then input to the seven-layer DNN as shown in Fig. 7. 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 [41]. 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.

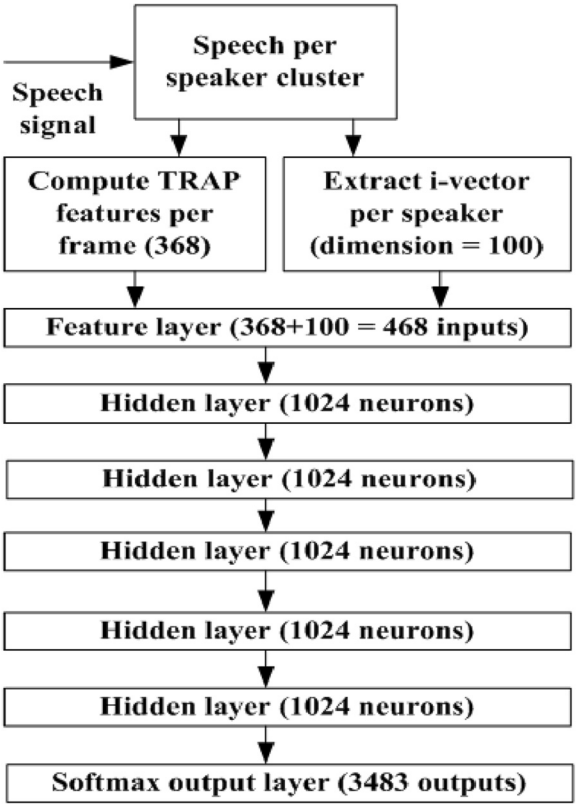

Fig. 7 Architecture of seven-layer DNN used with TRAP and i-vector features

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. Note that we did not use data from other test condition/conditions to perform feature normalization and adaptation per room (Fig. 7).

4.4 Algorithm used for decoding

The decoding algorithm is slightly different depending on whether we are doing utterance-based batch processing or full batch processing. For the maximum likelihood inverse filtering-based dereverberation (MLIFD) 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 the FMLLR transform from a single short utterance gives poor results (we need over 20 s 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 [42]. The modification is that each utterance corresponds to one speaker. There is no sub-segmentation of the utterance. Filterbank features in each speaker cluster are then normalized to zero. We also compute one 100-dimensional length-normalized i-vector per speaker [28].

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.

4.5 ROVER for combining the results

ROVER, developed by J. Fiscus of NIST [11], seeks to reduce word error rates for automatic speech recognition by exploiting differences in the nature of the errors made by multiple speech recognition systems. It works in two steps:

1. The outputs of several speech recognition systems are first aligned, and a single word transcription network (WTN) is built.
2. The second step consists of selecting the best scoring word (with the highest number of votes) at each node. The decision can also incorporate word confidence scores if these are available for all systems [11].

For the single-channel task, we use ROVER to combine the results of all seven systems to get the lowest possible word error rates (WERs). For the two-channel and eight-channel tasks, we, in place of applying any multi-microphone signal processing algorithms for doing the dereverberation, exploit ROVER to combine the results from all two and all eight channels, respectively, to get the best possible results.

5 Experiment results and discussion

The Kaldi recognizer [20] was used for training and recognition task. Experiments were carried out on the REVERB challenge 2014 corpora [32, 33, 35, 43], and results were reported on the all evaluation conditions for the one-channel, two-channel, and eight-channel tasks. Results reported here are only on the evaluation corpus. Development corpus was used for tuning the system parameters.

5.1 Results obtained 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 six different results using ROVER. Table 1 (one-channel task) shows the results for each feature and each room. The last row shows the results after combining results from all six features using ROVER. After ROVER, the average SimData WER is 10.0 % and average RealData WER is 27.1 %. All individual systems, except the $MMFB_p$, outperformed the baseline system, in terms of average WER, both in the SimData and RealData tasks. Among the individual systems, ITD-MFB performed the best in SimData, and in RealData, the RCGFB front-end provided lowest WER.

For the two-channel utterance-based recognition, the only difference from one-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 the one-channel case, since ROVER needs at least three 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 × 6 features). The order of combination in ROVER is the order of the rows in Table 1 (two-channel task). 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 two-channel task, except the $MMFB_p$ feature, all the features provided reduced WERs both in SimData and RealData tasks. The RCGFB and RMFB front-ends performed best in RealData, whereas ITD-MFB performed best in SimData.

In the eight-channel utterance-based recognition, for each feature parameter, we combine the results from all eight 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 eight channels with ROVER reduces the WER significantly. For combining recognition results from all the features (last row), there are 48 recognition outputs (8 channels × 6 features) to combine. The recognition outputs are combined in the same order as the rows in Table 1 (eight-channel task). 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 %. In the eight-channel task, except the $MMFB_p$ feature, all the systems outperformed the baseline, in terms of average WER, in SimData as well as in RealData tasks. Like the one-channel and two-channel tasks, in the eight-channel task, the ITD-MFB feature yielded the lowest WERs in SimData, and in RealData, the $MMFB_l$ feature performed the best.

Table 1 WER obtained using utterance-based batch processing for one-channel, two-channel, and eight-channel tasks

	SimData							RealData		
	Room 1		Room 2		Room 3		Avg.	Room 1		Avg.
	Near	Far	Near	Far	Near	Far		Near	Far	
One-channel task										
RCGFB	8.2	9.4	9.8	15	10.8	16.6	11.6	30.4	31.5	30.9
$MMFB_I$	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
Two-channel task										
RCGFB	8.4	9.5	10.1	15.2	11.1	17.1	11.9	31.4	32.4	31.9
$MMFB_I$	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
Eight-channel task										
RCGFB	8.1	9	9.1	14	10.3	15.3	11	27.9	28.7	28.3
$MMFB_I$	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

5.2 Results obtained 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 [42] 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 the i-vector for each speaker using the multi-taper MFCC features and used these i-vectors during training/recognition using other features. This strategy did not work well. Only the WER for $MMFB_I$ 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, the 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 (one-channel task). In the last column of Table 2 (one-channel task), we are combining results from seven different features

Table 2 WER obtained using full batch processing for one-channel, two-channel, and eight-channel tasks

	SimData							RealData		
	Room 1		Room 2		Room 3		Avg.	Room 1		Avg.
	Near	Far	Near	Far	Near	Far		Near	Far	
One-channel task										
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
$MMFB_I$	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
Two-channel task										
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
$MMFB_I$	8.8	10.4	10.5	17.9	11.8	19.5	13.2	29.5	28.8	29.1
$MMFB_p$	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	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
Eight-channel task										
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
$MMFB_I$	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

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 two-channel full batch processing, the only difference from one-channel processing is that for each room, we combine the results for the two channels using ROVER. As mentioned earlier, ROVER needs at least three inputs to reduce the WER significantly, and when we combine all the feature parameter outputs with ROVER, the combination, in this case, is for 14 different recognizers (2 channels × 7 features). The results are shown in the last row of Table 2 (two-channel task). 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 two-channel utterance-based processing.

For the eight-channel full batch processing, for each feature parameter, we combine the results from all eight 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 (eight-channel task) presents this combined result. Combining eight channels with ROVER reduces the WER significantly. For combining with ROVER, recognition results from all the features (last row in Table 2 (eight-channel task)), there are 56 recognition outputs (8 channels × 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 (eight-channel task). 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 two-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 %.

In full batch processing, in the one-, two- and eight-channel tasks, the ITD-MFB front-ends provided the lowest average WER in SimData, and in RealData, the MLIFD feature-based system yielded the lowest average WER.

6 Conclusions

In this work, we presented multiple cepstral or filterbank energy feature extraction technique-based speech recognition systems, and then we combined the recognition results with the help of ROVER (Recognizer Output Voting Error Reduction) to get the best possible recognition results, i.e., lowest WER. The feature extraction techniques chosen for the REVERB challenge task were convention mel-filterbank (MFB), multi-taper mel-filterbank (MMFB) with log and power-law nonlinearity, robust compressive gammachirp filterbank (RCGFB), robust MFB (RMFB), iterative deconvolution-based MFB (ITD-MFB), and maximum likelihood inverse filtering-based dereverberated (MLIFD) features. For speaker adaptation, we applied an i-vector-based speaker adaptation technique as it was found to boost the performance by approximately 2 % [13]. In the case of two- and eight-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 feature extractors except the MMFB with power function nonlinearity performed better in terms of WER both in SimData and RealData. The best results were obtained with the full batch processing and with the eight-channel task. For the eight-channel task, we obtained an average WER of 8.9 and 21.7 % on the SimData and RealData, respectively.

Competing interests

The authors declare that they have no competing interests.

Acknowledgements

The authors would like to thank the reviewers for their valuable comments which have enabled us to significantly improve the quality of the paper.

Author details

[1]CRIM, Montreal, Quebec, Canada. [2]ETS, Montreal, Quebec, Canada.

Received: 25 February 2015 Accepted: 8 June 2015
Published online: 19 June 2015

References

1. PA Naylor, ND Gaubitch, *Speech Dereverberation, Signals and Communication Technology Series*, 2010th edn. (Springer-Verlag, London, 2010)
2. Kumar K, Stern RM, Maximum-likelihood-based cepstral inverse filtering for blind speech dereverberation, *in Proc. of ICASSP* (Dallas, Texas, 2010)
3. Alam MJ, Gupta V, Kenny P, Dumouchel P, Use of multiple front-ends and i-vector based speaker adaptation for robust speech recognition, in *Proc. of REVERB Challenge* (Florence, Italy, 2014)
4. MJ Alam, P Kenny, P Dumouchel, D O'Shaughnessy, Robust feature extractors for continuous speech recognition, in *Proc. of EUSIPCO* (Lisbon, Portugal, 2014)
5. A Schwarz, C Heummer, R Maas, W Kellermann, Spatial diffuseness features for DNN-based speech recognition in noisy and reverberant environments, in *Proc. of ICASSP* (Brisbane, Australia 2015)
6. L Deng, J Li, J-T Huang, K Yao, D Yu, F Seide, M Seltzer, G Zweig, X He, J Williams, Y Gong, A Acero, Recent advances in deep learning for speech research at Microsoft, in *Proc. Int. Conf. Acoust., Speech, Signal Process* (Vancouver, Canada, 2013) p. 8604–8608
7. A-R Mohamed, G Hinton, G Penn, Understanding how deep belief networks perform acoustic modelling, in *Proc. Int. Conf. Acoust., Speech, Signal Process*, (2012) p. 4273–4276
8. T Yoshioka, A Ragni, MJF Gales, Investigation of unsupervised adaptation of DNN acoustic models with filter bank input, in *Proceed. of ICASSP* (Florence, Italy, 2014) p. 6344–6348. online: http://mi.eng.cam.ac.uk/~mjfg/yoshioka_ICASSP14.pdf
9. A Mohamed, G Dahl, G Hinton, Acoustic modeling using deep belief networks. IEEE. Trans. Audio. Speech. Lang. Process. **20**(1), 14–22 (2012)
10. G Dahl, D Yu, L Deng, A Acero, Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. IEEE. Trans. Audio. Speech. Lang. Process. **20**(1), 30–42 (2012)
11. JG Fiscus, A post-processing system to yield reduced error word rates: recognizer output voting error reduction (ROVER), in *IEEE Workshop on Automatic Speech Recognition and Understanding* (Santa Barbara, CA, 1997) p. 347–354
12. Y Tachioka, T Narita, FJ Weninger, S Watanabe, Dual system combination approach for various reverberant environments with dereverberation techniques, in *Proceed. of REVERB Challenge Workshop* (Florence, Italy, 2014). online: http://reverb2014.dereverberation.com/workshop/reverb2014-papers/1569886337.pdf
13. V Gupta, P Kenny, P Ouellet, T Stafylakis, I-vector-based speaker adaptation of deep neural networks for French broadcast audio transcription, in *Proc. of ICASSP* (Florence, Italy, 2014)
14. MJ Alam, T Kinnunen, P Kenny, P Ouellet, D O'Shaughnessy, Multitaper MFCC and PLP features for speaker verification using i-vectors. Speech Comm. **55**(2), 237–251 (2013)
15. MJ Alam, P Kenny, D O'Shaughnessy, Low-variance multi-taper mel-frequency cepstral coefficient features for speech and speaker recognition systems. Springer Cognit. Comput. J. **5**(4), 533–544 (2012)
16. S Dharanipragada, BD Rao, MVDR based feature extraction for robust speech recognition, in *Proc. of ICASSP* (Salt Lake Ciy, Utah, 2001) p. 309–312
17. D Slepian, HO Pollak, Prolate spheroidal wave functions, Fourier analysis and uncertainty - I. Bell. Syst. Tech. J. **40**, 43–63 (1960)
18. DJ Thomson, Spectrum estimation and harmonic analysis. Proc. IEEE. **70**(9), 1055–1096 (1982)
19. Alam J, Ouellet P, Kenny P, O'Shaughnessy D, Comparative evaluation of feature normalization techniques for speaker verification, in *Proc of NOLISP, LNAI 7015, 246–253* (Las Palmas, Spain, 2011)
20. D Povey, A Ghoshal, G Boulianne, L Burget, O Glembek, N Goel, M Hanneman, P Motlicek, Y Qian, P Schwarz, J Silovsky, G Stemmer, K Vesely, The Kaldi speech recognition toolkit, in *Proc. of ASRU* (Hawaii, 2011)
21. MJ Alam, P Kenny, D O'Shaughnessy, Robust feature extraction for speech recognition by enhancing auditory spectrum, in *Proc. of Interspeech* (Portland, Oregon, 2012)
22. MJ Alam, P Kenny, D O'Shaughnessy, Robust feature extraction based on an asymmetric level-dependent auditory filterbank and a subband spectrum enhancement technique, in *Digital Signal Processing*, 29 (2014) p. 147–157
23. Timo Gerkmann, Richard C. Hendrikes, Noise power estimation based on the probability of speech presence, in *Proc. IEEE WASPAA* (New York, 2011) p. 145–148
24. RC Hendriks, R Heusdens, J Jensen, MMSE based noise PSD tracking with low complexity, in *Proc. of IEEE ICASSP* (Dallas, Texas, 2010) p. 4266–4269
25. Kumar K, Raj B, Singh R, Stern RM, An iterative least-squares technique for dereverberation, in *Proc. of ICASSP* (Prague, Czech Republic, 2011)
26. Kumar K, Stern RM, Environment-invariant compensation for reverberation using linear post-filtering for minimum distortion, in *Proc. of ICASSP* (Las Vegas, Nevada, 2008)
27. N Dehak, P Kenny, R Dehak, P Dumouchel, P Ouellet, Front-end factor analysis for speaker verification. IEEE. Trans. Audio. Speech. Lang. Process. **19**(4), 788–798 (2011)
28. Kenny P, A small footprint i-vector extractor, in *Proc. of Odyssey Speaker and Language Recognition Workshop* (Singapore, 2012)

29. T Stafylakis, P Kenny, P Ouellet, J Perez, M Kockmann, P Dumouchel, *I-Vector/PLDA Variants for Text-Dependent Speaker Recognition* (CRIM, Montreal, 2013)
30. F Grezl, *TRAP-Based Probabilistic Features for Automatic Speech Recognition* (Dept. of Computer Graphics & Multimedia, Brno Univ of Technology, Brno, Czech Republic, Doctoral Thesis, 2007)
31. G Saon, H Soltau, D Nahamoo, M Picheny, Speaker adaptation of neural network acoustic models using i-vectors, in *Proc. of ASRU* (Olomouc, Czech Republic, 2013)
32. T Robinson, J Fransen, D Pye, J Foote, S Renals, WSJCAMO: a British English speech corpus for large vocabulary continuous speech recognition, in *Proc. of ICASSP* (Detroit, Michigan, 1995) p. 81–84
33. M Lincoln, I McCowan, J Vepa, HK Maganti, The multi-channel Wall Street Journal audio visual corpus (MC-WSJ-AV): specification and initial experiments, in *Proc. of ASRU* (Cancun, Mexico, 2005) p. 357–362
34. FJ Weninger, S Watanabe, J Le Roux, J Hershey, Y Tachioka, JT Geiger, BW Schuller, G Rigoll, The MERL/MELCO/TUM system for the REVERB challenge using deep recurrent neural network feature enhancement, in *Proc. of REVERB Challenge Workshop* (Florence, Italy, 2014)
35. K Kinoshita, M Delcroix, T Yoshioka, T Nakatani, E Habets, R Haeb-Umbach, V Leutnant, A Sehr, W Kellermann, R Maas, S Gannot, B Raj, The REVERB challenge: a common evaluation framework for dereverberation and recognition of reverberant speech, in *Proceedings of the WASPAA* (New Paltz, NY, 2013)
36. K Vesel'y, A Ghosal, L Burget, D Povey, Sequence discriminative training of deep neural networks, in *Proc. of Interspeech* (Lyon, France, 2013) p. 2345–2349
37. RO Duda, PB Hart, *Pattern Classification and Scene Analysis* (Wiley, New York, 1973)
38. MJF Gales, Semi-tied covariance matrices for hidden Markov models. IEEE. Trans. Speech. Audio. Process. **7**(3), 272–281 (1999)
39. MJF Gales, Maximum likelihood linear transformations for HMM-based speech recognition. Proc. Comput. Speech. Lang. **12**, 75–98 (1998)
40. TN Sainath, B Kingsbury, A-R Mohamed, GE Dahl, G Saon, H Soltau, B Beran, AY Aravkin, B Ramabhadran, Improvements to deep convolutional neural networks for LVCSR, in *Proceed. of Automatic Speech Recognition and Understanding Workshop* (Olomouc, Czech Republic, 2013) p. 315–320,
41. D Garcia-Romero, CY Espy-Wilson, Analysis of i-vector length normalization in speaker recognition systems, in *Proc. of Interspeech* (Florence, Italy, 2011)
42. V Gupta, G Boulianne, P Kenny, P Ouellet, P Dumouchel, Speaker diarization of French broadcast news, in *Proc. of ICASSP* (Las Vegas, 2008) p. 4365–4368
43. D Paul, B Baker, M Janet, The design for the Wall Street Journal-based CSR corpus, in *Proc. of HLT* (Harriman, NY, 1992) p. 357–362