Investigation of Speech Separation as a Front-End for Noise Robust Speech Recognition

Arun Narayanan, *Student Member, IEEE*, and DeLiang Wang, *Fellow, IEEE*

***Abstract*—Recently, supervised classification has been shown to work well for the task of speech separation. We perform an in-depth evaluation of such techniques as a front-end for noise-robust automatic speech recognition (ASR). The proposed separation front-end consists of two stages. The first stage removes additive noise via time-frequency masking. The second stage addresses channel mismatch and the distortions introduced by the first stage; a non-linear function is learned that maps the masked spectral features to their clean counterpart. Results show that the proposed front-end substantially improves ASR performance when the acoustic models are trained in clean conditions. We also propose a diagonal feature discriminant linear regression (dFDLR) adaptation that can be performed on a per-utterance basis for ASR systems employing deep neural networks and HMM. Results show that dFDLR consistently improves performance in all test conditions. Surprisingly, the best average results are obtained when dFDLR is applied to models trained using *noisy* log-Mel spectral features from the multi-condition training set. With no channel mismatch, the best results are obtained when the proposed speech separation front-end is used along with multi-condition training using log-Mel features followed by dFDLR adaptation. Both these results are among the best on the Aurora-4 dataset.**

***Index Terms*—Aurora-4, deep neural networks, feature mapping, robust ASR, time-frequency masking.**

I. Introduction

Automatic speech recognition systems are finding applications in an array of tasks. Although these systems have become fairly powerful, the inherent variability of an acoustic signal can still pose challenges. The sources of variability are many, ranging from speaker idiosyncrasies to recording channel characteristics. A widely studied problem is the variability caused by background noise; ASR systems that work well in clean conditions suffer from a drastic loss of performance in the presence of noise. In most cases, this is caused due to the mismatch in training and testing conditions. Techniques have been developed to handle this *mismatch problem*. Some focus on feature extraction using robust features like RASTA [19] and AFE [13], or feature normalization [3]. In *feature-based* methods, an enhancement algorithm modifies noisy features so that they more closely follow the distributions of the training data [32], [9]. Alternatively, one could train models using features extracted from multiple 'noisy' conditions. Feature-based techniques have the potential to generalize well, but do not always produce the best results. In *model-based* approaches, the ASR model parameters are adapted to match the distribution of noisy or enhanced features [15], [45]. Model-based methods work well when the underlying assumptions are met, but typically involve significant computational overhead [45]. The best performances are usually obtained by combining feature-based and model-based approaches (e.g., an uncertainty decoding framework [8]).

The current work focuses on feature-based methods. Such methods can be classified into two groups depending on whether or not they use stereo training data[1]. When stereo data is unavailable, prior knowledge about speech and/or noise is used to perform feature enhancement. Examples include spectral reconstruction based missing feature methods [32], direct masking methods described in [18], [28], and feature enhancement methods [1]. When stereo data is available, feature mapping methods like SPLICE [7] and recurrent neural networks [25] have been used. Even though generalization to unseen conditions is a challenge when using stereo training data, such methods work well when the training and test noises are similar. We can also generate psuedo-clean signals directly from noisy signals for training such stereo systems [10].

Stereo training data is also used by supervised classification based speech *separation* algorithms [33], [36], [22], [17], [43]. Such algorithms typically estimate the ideal binary mask (IBM)–a binary mask defined in the time-frequency (T-F) domain that identifies speech dominant and noise dominant T-F units [40]. Our recent work extends this approach to estimate the ideal ratio mask (IRM) [29]. The IRM represents the ratio of speech to mixture energy at each T-F unit [38]. Results show that using the estimated IRM as a front-end to perform feature enhancement significantly improves ASR performance, and works better than using the estimated IBM. Only additive noise is addressed in [29]. In this work, we address both additive noise and convolutional distortion due to recording channel mismatch. They are dealt with in two separate stages: Noise

Manuscript received June 21, 2013; revised October 17, 2013; accepted February 06, 2014. Date of publication February 12, 2014; date of current version February 20, 2014. This work was supported in part by the Air Force Office of Scientific Research under Grant (FA9550-08-1- 0155) and in part by the Ohio Supercomputer Center. A preliminary version of this work was presented at ICASSP-2013 [29]. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Brian Kingsbury.

A. Narayanan is with the Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, 43210-1277 USA (e-mail: narayaar@cse.ohio-state.edu).

D. L. Wang is with the Department of Computer Science and Engineering and Center for Cognitive and Brain Sciences, The Ohio State University, Columbus, OH, 43210-1277 USA (e-mail: dwang@cse.ohio-state.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASLP.2014.2305833

[1]By stereo we mean noisy and the corresponding clean signals.

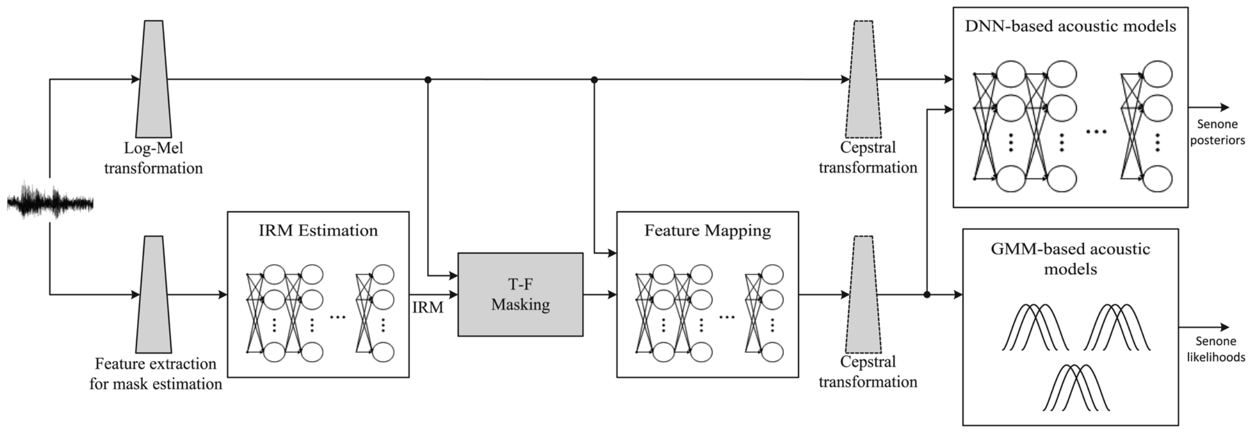


Fig. 1. Block diagram of the proposed speech separation front-end. The different stages of the front-end, namely time-frequency masking and feature mapping, and their corresponding inputs are shown in the figure. The ASR module may be implemented either using GMMs or DNNs.

removal followed by channel compensation. Noise is removed via T-F masking using the IRM. To compensate for channel mismatch and the errors introduced by masking, we learn a non-linear mapping function that undoes these distortions. Estimation of the IRM and learning the mapping function are both accomplished using deep neural networks (DNNs) [21].

In feature-based robust ASR methods, the enhanced features are passed on to the back-end ASR system for final decoding. Acoustic modeling in ASR has traditionally been done using Gaussian mixture models (GMMs). Our first set of evaluations uses GMM based acoustic models to measure how the proposed separation front-end affects performance. Next, we study how the front-end performs with DNN based acoustic models, which are now the state-of-the-art [5]. We also propose a feature adaptation technique for DNN-based systems. Unlike previous adaptation techniques like feature discriminant linear regression (FDLR) [35], the proposed method can be used on a per-utterance basis. As shown in our results, feature adaptation can partly address the generalization issue of supervised learning systems like the DNNs.

The rest of the paper is organized as follows. The components of our system are described in detail in the next section. Evaluation results are presented in Section III. We discuss in Section IV our results and future research directions.

II. System Description

A block diagram of the proposed system is shown in Fig. 1. The IRM is estimated using a group of features extracted specifically for the purpose. As shown, feature mapping is done using both the log-mel transformed noisy features and the features obtained after masking. The mapped feature may or may not be transformed to the cepstral domain, depending on the type of acoustic models used. The acoustic models are implemented either using GMMs or DNNs. The different stages of the system are described in detail below.

A. Time-Frequency Masking

We perform T-F masking in the mel-frequency domain unlike some of the recent systems that operate in the gammatone feature domain [17], [42]. When using conventional ASR features like mel-frequency cepstral coefficients (MFCC), this lets us bypass the signal resynthesis step after masking (e.g., see [18]). Since the mel-domain lies in the pathway of MFCC/log-mel feature extraction, masking only adds a single step (point-wise matrix multiplication). Further, the number of frequency bands used in the mel-domain (20–26) is typically lower than those in the gammatone domain (32–64). When using subband classifiers for mask estimation [17], this reduces the number of classifiers that needs to be trained.

To obtain the mel-spectrogram of a signal, it is first pre-emphasized and transformed to the linear frequency domain using a 320 channel fast Fourier transform (FFT). A 20 msec Hamming window with an overlap of 10 msec is used. The 161-dimensional spectrogram is then converted to a 26-channel mel-spectrogram using triangular mel filters that span frequencies in the range [50 Hz, 7000 Hz]. The RASTAMAT toolkit is used to perform these operations [12].

We use DNNs[2] to estimate the IRM as they show good performance [43] and training using stochastic gradient descent scales well compared to other nonlinear discriminative classifiers, such as SVMs, as the size of the dataset increases [2]. Below we describe the three components of this supervised learning method: the IRM, the features, and the learning strategy.

1) Target Signal: The ideal ratio mask is defined as the ratio of the clean signal energy to the mixture energy at each time-frequency unit. Assuming speech and noise are uncorrelated, the mixture energy can be approximated as the sum of clean and noise energy. This limits the values in the IRM in the range [0, 1]. The IRM can now be defined in terms of the instantaneous signal-to-noise ratio (SNR) at each T-F unit:

$$\begin{aligned} IRM(t,f) &= \frac{10^{(SNR(t,f)/10)}}{10^{(SNR(t,f)/10)}+1}, \\ SNR(t,f) &= 10\log_{10}(\mathbf{X}(t,f)/\mathbf{N}(t,f)). \end{aligned} \tag{1}$$

Here, $\mathbf{X}(t,f)$ and $\mathbf{N}(t,f)$ denote the instantaneous speech and noise energy, respectively, at time frame t and frequency channel f. $SNR(t,f)$ denotes the instantaneous SNR in dB. As can be seen, there is a one-to-one correspondence between $IRM(t,f)$ and $SNR(t,f)$. In our earlier work, we observed that estimating a transformed version of the SNR, instead of the IRM directly, works better [29]. This is probably because the

[2]We call a neural network with a single hidden layer a multi-layer perceptron (MLP), and with more than one hidden layer a DNN.

transformation gives a better representation to the SNR values in the range close to 0 dB. The sigmoidal transformation that we use is of the following form:

$$d(t,f) = \frac{1}{1+\exp(-\alpha(SNR(t,f)-\beta))}. \tag{2}$$

$d(t,f)$ denotes the desired target that is used during training. α and β are the parameters that control the slope and the bias, respectively, of the sigmoid. Indirectly, they control the range of SNR to be focused during training. Following our preliminary work, we set α to roughly have a 35 dB SNR span[3] centered at β, which is set to -6 dB (see [41]). During testing, the values output from the DNN are mapped back to their corresponding IRM values using the inverse of Eqs. (2) and (1). The resulting mask is used for T-F masking.

2) Features: Feature extraction is performed both at the fullband (signal) and the subband level. To extract subband features, the original fullband signal is filtered using a 26-channel mel filterbank that spans frequencies from 50 Hz to 7000 Hz. The filterbank is implemented using sixth order Butterworth filters, with the same cutoff frequencies used when transforming the signal to its mel-spectrogram. Note that the output of filterbank is the continuous subband signals sampled at the same frequency as the input signal. Subband features are extracted following the same processing steps as for fullband features, but using the subband filtered signals.

We use a combination of features, similar to that used in [42] for IBM estimation. The following base features are extracted:

- 31 dimensional MFCCs, derived from a 64 channel mel-spectrogram. A frame length of 20 msec and an overlap of 10 msec are used. MFCC features are extracted for both subband and fullband signals.
- 13 dimensional RASTA filtered PLP cepstral coefficients (RASTA-PLPs). A frame length of 20 msec and an overlap of 10 msec are used. RASTA-PLP features are extracted for both subband and fullband signals.
- 15 dimensional amplitude modulation spectrogram (AMS) features. AMS features are extracted only for subband signals. Features are extracted after downsampling the subband signals to 4000 Hz. A frame length of 32 msec with 22 msec overlap (corresponding to 10 msec hop size) is used.

Using these base features, the following derived features are obtained:

- *Fullband features*: The fullband features are derived by splicing together fullband MFCCs and RASTA-PLPs, along with their delta and acceleration components, and subband AMS features. The dimensionality of this feature is 522 $((31 \times 3) + (13 \times 3) + (15 \times 26))$.
- *Subband features*: The subband features are derived by splicing together subband MFCCs, RASTA-PLPs, and AMS features. Delta and acceleration components are added to MFCCs and RASTA-PLPs; temporal and spectral deltas are added to the AMS features. The dimensionality of this feature is 177 $((31 \times 3) + (13 \times 3) + (15 \times 3))$. Note that subband features are obtained for each of the 26 subband signals.

Global mean and variance normalization and a second order moving average smoothing [3] is applied to both fullband and subband features to improve robustness.

3) Supervised learning: IRM estimation is performed in two stages. In the first stage, multiple DNNs are trained using fullband and subband features. The final estimate is obtained using an MLP that combines the output of the fullband and the subband DNNs.

The fullband DNN is trained using the fullband features and learns to estimate the desired target (see Eq. 2) corresponding to the 26 frequency channels. The DNN uses 2 hidden layers, each with 1024 nodes. The output layer consists of 26 nodes corresponding to the 26 mel-frequency channels. Sigmoid activation function is used for both hidden and output nodes. The weights of the hidden units are initialized layer-by-layer using restricted Boltzmann machine (RBM) pretraining. A learning rate of 0.01 is used for the first hidden layer; it is set to 0.1 for the second hidden layer. A momentum of 0.9, and a weight decay of 0.0001 are used. Each layer is pretrained for 50 epochs. The output layer is then added to the network and the weights of the full network are fine-tuned using supervised backpropagation. While fine-tuning, we use adaptive gradients with a global learning rate of 0.05 [11]. Momentum is initially set to 0.5 and then increased to 0.9 after the network trains for half the total number of epochs, which is set to 100. The cross-entropy error criterion is used. Throughout the training procedure we use minibatch gradient descent with the minibatch size set to 2048.

The subband DNNs are individually trained for each of the 26 frequency channels. Each of these DNNs consists of 2 hidden layers just like the fullband DNNs, but has only 200 nodes per layer. The output layer has a single node. The training schedule is the same as for training the fullband DNN.

Although the outputs from the fullband and the subband DNNs strongly correlate, we expect both these networks to learn useful information not learned by the other. The fullband DNNs would be cognizant of the overall spectral shape of the IRM and the information conveyed by the fullband features, whereas the subband DNNs are expected to be more robust to noise occurring at frequencies outside their passband. Therefore, the final prediction is made by combining the outputs of the fullband and the subband classifiers. An additional purpose of this combination is to utilize the information contained in the neighboring T-F units surrounding each unit in an explicit fashion–the fullband and the subband DNNs estimate the output at each unit mostly using temporally local features. We train a simple MLP to perform this combination. This MLP has 26 output nodes, corresponding to the 26 frequency channels, and 512 hidden nodes. Its input is created by splicing together the outputs of the fullband and the subband DNNs from five leading and five trailing frames surrounding the current frame for which the final output is to be estimated. This results in a 572 dimensional input feature $(26 \times 11 \times 2)$. The MLP is trained for 250 epochs to minimize the cross-entropy error, as before. Instead of using adaptive gradients, we use a more conventional method to set the learning rate–it is linearly decreased from 1 to 0.001 over the training time. It is worth mentioning that in

[3]We define SNR span as the difference between the instantaneous SNRs corresponding to the desired target values of 0.95 and 0.05.

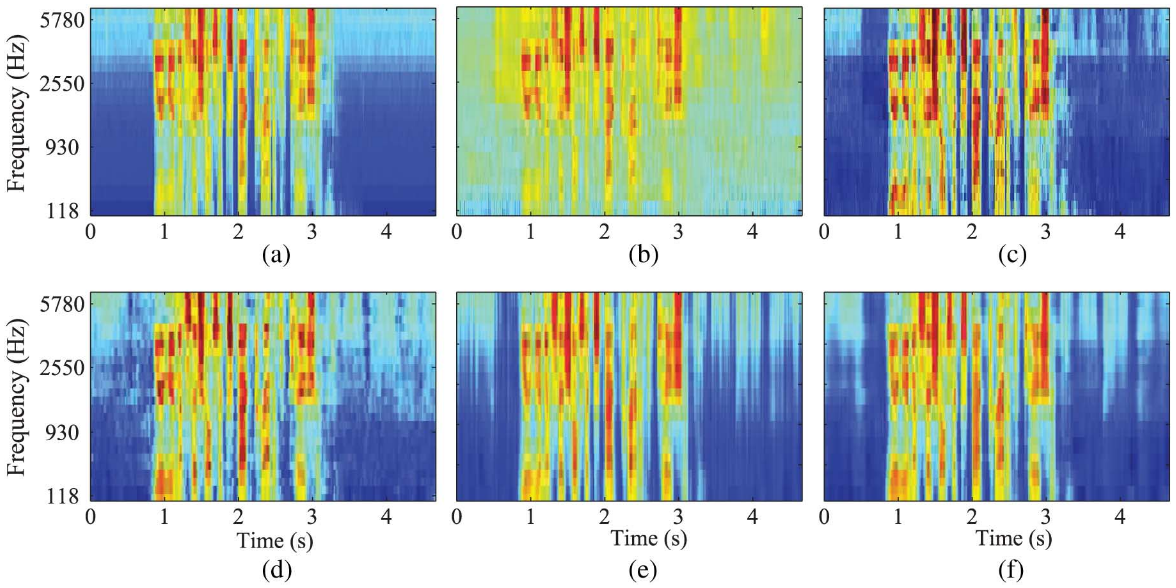


Fig. 2. (Color online) Example of T-F masking. (a) Mel spectrogram of a clean signal from the Aurora-4 corpus. (b) Mel spectrogram of the same signal mixed with babble noise. (c) Map of the true instantaneous SNRs, expressed in dB. (d) Instantaneous SNRs estimated using subband features and the corresponding DNNs. The mean absolute SNR estimation error for this mask is 2.9 dB. (e) Instantaneous SNRs estimated using fullband features. SNR estimation error for this mask is 2.5 dB. (e) Instantaneous SNR estimates obtained after combining the masks in (d) and (e). It can be noticed that the mask in (f) is smoother than those in (d) and (e). SNR estimation error for this mask is 2.1 dB. Note that the SNR estimates are rounded to the range $[-15, 10]$ dB before calculating the mean absolute error.

preliminary experiments not reported in the paper, we observed that the above combination worked the best compared to using the output of only the fullband or the subband classifier as input to the MLP. Further, on a noisy development set, the 'combined' mask worked 11% (relative) better than the subband masks and 5% (relative) better than the fullband masks, in terms of averaged word error rates (WER), when using GMM based acoustic modes trained in clean conditions.

All networks are trained using the noisy subset of the training set of the Aurora4 corpus. Of the 2676 utterances in this subset, 2100 are used for training and the remaining for cross validation; the network that gives the least mean squared error on the cross validation set is chosen in each case.

Fig. 2 shows an example of T-F masking. As can be seen, the mask estimated by the fullband and the subband DNNs look similar, but the one estimated by the fullband DNN is spectrally smoother. As expected, the output obtained by combining the two masks results in a temporally smooth mask. The local SNR estimation error is also lower by about 0.4 dB for the combined mask.

B. Feature mapping

Time-frequency masking only addresses additive noise. As we will show in Section III-A, even after T-F masking, channel mismatch can still significantly impact performance. This happens for two reasons. Firstly, our algorithm learns to estimate the ratio mask using mixtures of speech and noise recorded using a single microphone. Secondly, because channel mismatch is convolutional, speech and noise, which now includes both background noise and convolutive noise, are clearly not uncorrelated. So even if the training set used to estimate the IRM included examples with channel distortions, the masking function will still be ill-defined: The residual noise obtained after subtracting the clean signal from a noisy channel-mismatched signal will have speech-like characteristics. While recent studies in speech perception have shown that *binary* masking can be used to improve intelligibility of noisy reverberant mixtures [34], it is not straightforward to set the target in our case since we assume no access to the clean signals recorded using every microphone in our dataset. Therefore, we use an alternative strategy to deal with channel mismatch that directly maps the noisy/masked signals to their 'clean' version. The clean version here corresponds to the signals in the clean datasets of the corpus.

Feature mapping has already been used for enhancing noisy features for speech recognition [7], [25], [46] and speech separation [44]. The goal of feature mapping in this work, however, is to learn spectro-temporal correlations that exist in speech to undo the distortions introduced by unseen microphones and the first stage of our algorithm. Our feature mapping function operates in the log-mel spectral domain, unlike the earlier works which operate in the cepstral or other feature domains. The target, the features, and the learning strategy are described below.

1) Target Signal: The goal of mapping is to transform the input spectra to its clean log-mel spectra. This sets the target to be the clean log-mel spectrogram (LMS). The 'clean' LMS here corresponds to those obtained from the clean signals recorded using a single microphone in a single filter setting. Since we use Aurora-4 for our experiments, this corresponds to the data in the clean training set of the corpus recorded using a Sennheiser microphone and processed by a P.341 filter. Instead of using the LMS directly as the target, we apply a linear transform to limit the target values to the range [0, 1] to allow us to use the sigmoidal transfer function for the output layer of the DNN. Experimentally, we found this to make learning easier compared to using linear output neurons and unscaled targets. The transformation, which is scale invariant, has the following form:

$$\mathbf{X}_d(t,f) = \frac{\ln(\mathbf{X}(t,f)) - \min(\ln(\mathbf{X}(\cdot,f)))}{\max(\ln(\mathbf{X}(\cdot,f))) - \min(\ln(\mathbf{X}(\cdot,f)))}. \tag{3}$$

Here, $\mathbf{X}_d(t,f)$ is the desired target while learning, $\ln(\mathbf{X}(t,f))$ corresponds to the log-mel spectral energy at time frame t and frequency channel f. The $\min$ and $\max$ operations are over the entire training set. During testing, the output of the DNN is mapped back to the dynamic range of the utterances in training set using the inverse of the above transformation. It should be noted that even though $\min$ and $\max$ operations are typically affected by outliers, it worked well for our task as the training set was created in controlled conditions.

2) Features: As features we use both the noisy and the masked LMS. This was found to work better than using either the noisy LMS or the masked LMS. Since the DNNs that estimate the IRM are trained using signals recorded using a single microphone, it is possible that T-F masking introduces unwanted distortions in the presence of a channel mismatch. We believe that using both the noisy and masked LMS as input helps the DNN learn a mapping function robust to such distortions. We also append the log-mel features with their delta components. The final feature at a particular time frame is obtained by splicing together the spectrogram and the deltas of the current frame with those of five leading and trailing frames. This results in a 1144 $((26 \times 2) \times 11 \times 2)$ dimensional input. The features are global mean-variance normalized.

3) Supervised learning: For learning the mapping function, we use a three-layer DNN with 1024 nodes in each layer. The output layer consists of 26 nodes corresponding to the number of frequency channels. Unlike the DNNs used for IRM estimation, the hidden layers of the DNN for this task use rectified linear units (ReLU) [27]. ReLU has been shown to work well in a number of vision tasks [24], and has also been used in ASR [4]. We found ReLU to work at least as well as the sigmoidal units even without any pretraining; they also converged faster. The output layer uses sigmoid activations. The learning rate is linearly decreased from 1 to 0.001 over 50 epochs. A momentum of 0.5 is used for the first 10 epochs after which it is set to 0.9. Weight decay is set to 0.00001. The network is trained to minimize the squared error loss. We use all the utterances, 7138 of them, in the multi-condition training set of Aurora4 for training. A cross-validation set, which was created by randomly sampling 100 utterances in each condition from the development set of Aurora4 is used for early stopping–we chose the model that gave the least mean squared error on the cross-validation set during training.

Fig. 3 shows an example of how feature mapping reduces channel distortion. It can be clearly seen that, even though T-F masking removes some noise, the masked LMS still contains distortions. The LMS obtained after feature mapping looks much closer to the clean spectrogram; feature mapping is able to reconstruct the high frequency spectral components that are masked in the original signal. It should be noted that the microphone used to record the example shown in Fig. 3(b) is not used while training the DNN; in other words, the DNN does a reasonable job at generalizing to unseen microphones.

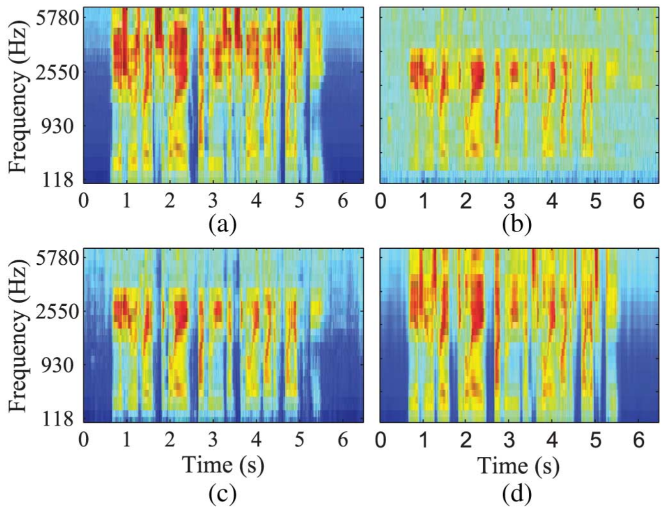


Fig. 3. (Color online) Example of feature mapping. (a) Mel spectrogram of the clean signal recorded using a Sennheiser microphone and processed with P.314 filter. (b) Mel spectrogram of the signal recorded using an alternative microphone and mixed with babble noise. The microphone attenuates the high frequency components of the signal. (c) The mel spectrogram after T-F masking. Noise has largely been removed but the high frequency components are still attenuated. (d) The mel spectrogram after feature mapping. As can be seen, the high frequency components are reconstructed reasonably well.

C. Acoustic modeling

The acoustic models are trained using the Aurora-4 dataset [30]. Aurora-4 is a 5000-word closed vocabulary recognition task based on the Wall Street Journal database. The corpus has two training sets, clean and multi-condition, both with 7138 utterances. The clean training set consists of signals recorded using a Sennheiser microphone and processed using a P.314 filter. The multi-condition set consists of both clean and noisy signals (SNR between 10 and 20 dB) recorded either using the Sennheiser microphone or one of 18 other microphones. The test set consists of 14 subsets, each with 330 utterances. Sets 1 to 7 and 8 to 14 contain utterances with and without channel mismatch, respectively. Sets 2 to 7 and 9 to 14 are noisy, whereas 1 and 8 are clean. Six noise types are considered: car, babble, restaurant, street, airport, and train. The noise types are the same as those used in the training set, but they are mixed at lower SNRs (SNR between 5 and 15 dB). The microphones used to create mixtures in sets 8 to 14 are different from those used in the training set.

We explore two types of acoustic models: the traditional GMM-HMM based systems and the more recent DNN-HMM based hybrid systems [5]. They are described in detail below.

1) Gaussian Mixture Models: The GMM based systems are trained using 12th order MFCCs, which are obtained after applying the discrete cosine transform and liftering to the LMS, along with their delta and acceleration coefficients. The features are mean and variance normalized at the sentence level. Normalizing the variance in addition to the mean significantly improves performance when using cepstral features [28]. The acoustic models consist of state-tied cross word triphones modeled as a 3-state HMM. The observation probability is implemented as a 16-component GMM with diagonal covariances. The training recipe is adapted from [39]. Based on the tree pruning parameters used for state-tying, the final models have 3319 tied-states or senones [5]. The HMMs and the GMMs are initially trained using the clean training set. The clean models are then used to initialize the multi-condition models; both clean and multi-condition models have the same structure and differ only in transition and observation probability densities.

2) Deep Neural Networks: The DNN based models are derived from the clean GMM-HMM system. We first align the clean training set to obtain senone labels at each time-frame for all utterances in the training set. DNNs are then trained to predict the posterior probability of senones using either clean features or features extracted from the multi-condition set.

We experimented with DNNs trained using cepstral and log-mel features. For either of those features we look at the following alternatives:

- Features extracted from the clean training set. While we expect models trained using clean features to generalize poorly, this will clearly show how effective the proposed speech separation front-end is. The input is formed by adding the cepstral/log-mel features with their delta and acceleration coefficients, and splicing together the features of 11 contiguous frames (5 leading and trailing frames surrounding each frame). This results in a 429-dimensional representation when the features are defined in the cepstral domain, and an 858-dimensional representation when they are defined in the log-mel domain.
- Features extracted from the multi-condition training set. These features are created just like the clean features, but since they are extracted from the multi-condition set they are expected to generalize much better. They have the same dimensionality as the clean features.
- Features generated after speech separation (masking followed by feature mapping) concatenated with noisy features from the multi-condition training set. To generate these features we perform speech separation on all utterances from the multi-condition training set using the models described in the previous sections. We found that adding the noisy features to the features after separation results in better performance compared to using only the latter. This trend is similar to that found while learning the feature mapping function (see Section II-B). The dimensionality of this feature is 858 when it is defined in the cepstral domain, and 1716 when defined in the log-mel domain.

The cepstral features are mean and variance normalized at the sentence level, like before. The log-mel features are mean normalized at the sentence level; an additional global variance normalization is also applied as suggested in [48].

The DNNs trained using cepstral features have an architecture similar to that described in [5]. It consists of 5 hidden layers, each with 2048 hidden units. The output layer has 3319 nodes. All hidden nodes use sigmoidal activation function. The output layer uses the softmax function. The weights of the hidden layers are initialized using RBM pretraining. The first hidden layer is pretrained for 100 epochs, the subsequent layers are pretrained for 35 epochs each. The learning rate for the first layer is set to 0.004. It is set to 0.01 for the remaining layers. The momentum is linearly increased from 0.5 to 0.9 over 10 epochs, and kept at 0.9 for the remaining epochs. Weight decay is set to 0.0001. After pretraining, the network is fine-tuned for 20 epochs using a learning rate of 0.08 for the first 10 epochs and 0.002 for the remaining ones. The momentum is set in the same way as in pretraining. The cross-entropy error criterion is used. Minibatch gradient descent with a batch size of 256 is used all through. A development set is used for early stopping based on the frame-level classification error. The network always stopped within the last 2-3 epochs; therefore early stopping did not affect performance.

The DNNs in the log-mel domain are trained using the recipe in [48]. The networks have 7 layers, each with 2048 nodes. The output layer has 3319 nodes, as before. For weight initialization, RBM pretraining is used. The first layer is pretrained for 35 epochs, and the remaining layers for 8 epochs. A constant learning rate and weight decay of 0.004 and 0.0001, respectively, are used. The minibatch size is set to 256. The network is fine-tuned based on the cross-entropy error criterion for 25 epochs with no early stopping. The minibatch size is set to 256 for the first 5 epochs. For the remaining 20 epochs, it is set to 1024. The learning rate is set to 0.08 for the first 5 epochs, 0.32 for the next 10 epochs and then reduced to 0.008 for the final 10 epochs. Momentum is fixed to 0.9 in the pretraining and fine-tuning stages.

It should be pointed out that training recipes have not been fully optimized, mainly due to the huge parameter search space and the amount of training/cross validation time required to evaluate each setting. A better way to choose parameters, e.g. using a Bayesian search strategy [4], could potentially be used in the future. Nevertheless, the performance trends obtained using the above networks are in line with those reported in other studies in the literature.

The senone posteriors estimated by the DNN are converted to likelihoods by normalizing using their priors calculated from the training set. The likelihoods are then passed on to the HMMs for decoding.

For both GMM-HMM and DNN-HMM systems, we use the CMU pronunciation dictionary and the standard bigram language model during decoding. The ASR systems are implemented using the HTK [47], which is adapted to also function as a hybrid system.

D. Diagonal Feature Discriminant Linear Regression

Feature adaptation techniques for DNN-HMM systems have largely been aimed at developing speaker-specific transforms. This generally requires a speaker-specific adaptation set with multiple utterances. While most studies try to reuse the adaptation methods developed for GMM-HMM systems like fMLLR (feature space maximum likelihood linear regression) and VTLN (vocal tract length normalization) with limited success [35], [48], DNN-specific adaptations like feature discriminant linear regression have also been proposed [35]. FDLR essentially learns an affine transformation of the features, like fMLLR, by minimizing the cross-entropy error of the labeled adaptation data. When FDLR is used on a per-utterance basis, it will likely cause overfitting as the number of parameters to learn is almost in the same range of, and sometimes greater than, the number of labeled examples available. Here, we propose a constrained version that gives us performance improvements in almost every test condition.

The motivation for developing dFDLR is to address the problem of generalization to unseen microphone conditions in our dataset, which is where the DNN-HMM systems perform the worst. dFDLR is a semi-supervised feature adaptation technique. To apply dFDLR, we first obtain an initial senone-level labeling for our test utterances using the unadapted models. Features are then transformed to minimize the cross-entropy error in predicting these labels. The adaptation takes the following form:

$$\widehat{\mathbf{O}}_t(f) = w_f \cdot \mathbf{O}_t(f) + b_f; t = 1 \ldots T, f = 1 \ldots F,$$
$$\min \sum\nolimits_t E(s_t, \mathbf{D}_{out}(\widehat{\mathbf{O}}_{t-5} \ldots \widehat{\mathbf{O}}_{t+5})). \qquad (4)$$

Here, $\widehat{\mathbf{O}}_t(f)$ is the feature obtained after transforming $\mathbf{O}_t(f)$, the original features. t indexes the time frame and f the feature dimension. T and F are the total number of time frames and features dimensions, respectively. E is the error function to be minimized, which is cross-entropy in our case. s_t is the senone label obtained in the first step of the adaptation process. $\mathbf{D}_{out}(\mathbf{O})$ is the output of the DNN when its input is $\mathbf{O}$. w_f and b_f are the parameters of the transformation. As can be seen, dFDLR simply scales and offsets the unadapted feature. When the input to the DNN consists of multiple feature streams, like in the case of DNNs that use features obtained after speech separation concatenated with noisy features, separate transforms are learned for each stream.

The parameters can easily be learned within the DNN framework by adding a layer between the input layer and the first hidden layer of the original DNN. We initialize w_f and b_f, for $f = 1 \ldots F$, to 1 and 0, respectively, and run the standard back-propagation algorithm for 10 epochs to learn these parameters. During backpropagation, weights of the original hidden layers are kept unchanged and only w_f and b_f are updated. Note that the parameters are tied across the 11 frames that are spliced together to create the input feature for the DNN. The parameter matrix and their updates have a form similar to the block diagonal weight matrices used by FDLR, but every block is additionally constrained to be diagonal in dFDLR. The learning rate and the momentum are set to 0.1 and 0.9, respectively, while learning these parameters. Before each update, the gradients are averaged across the frames of the sentence for which adaptation is being performed.

III. Evaluation Results

A. Gaussian Mixture Models

The averaged WER obtained using the GMM-HMM systems are tabulated in Table I. Using the models trained in clean conditions we obtain an average WER of 32.8% across the four conditions. Using the advanced front-end (AFE), an ETSI standardized feature representation [13], performance improves to 29.2 percent. AFE has a feature enhancement module to counter noise to an extent. Using a DNN that maps noisy log mel features directly to clean features gives a significant improvement, reducing the WER to 21.4 percent. Using T-F masking alone results in an average WER of 24.5 percent, but performs better than only using feature mapping when there is no channel mismatch. Using the proposed front-end with both masking and feature mapping, as described in Section II, gives the best average performance of 20.6 percent, a 37.2% relative improvement over the noisy baseline and a 29.5% improvement over the performance obtained using AFE features.

TABLE I
Word error rates on the Aurora-4 corpus using the GMM-HMM systems. The columns Clean, Noisy, Clean + Channel, and Noisy + Channel correspond to the WER averaged on test sets 1, 2 to 7, 8, and 9 to 14, respectively. The best performance in each condition is marked in bold. Results obtained using VTS-based model adaptation is also shown

System	Clean	Noisy	Clean + Channel	Noisy + Channel	Average
Clean Training					
Noisy	9.1	27.0	22.9	44.3	32.8
AFE	**9.0**	23.2	29.9	38.4	29.2
Feature mapping	9.8	16.3	14.3	29.6	21.4
T-F masking	9.4	15.3	23.1	36.4	24.5
+ feature mapping	9.7	**15.2**	**14.1**	**28.9**	**20.6**
Multi-condition Training					
Noisy	10.6	17.2	17.7	31.8	23.0
AFE	**10.3**	18.4	20.0	30.4	23.1
Feature mapping	11.7	15.9	14.7	27.5	20.5
T-F masking	10.7	**14.3**	20.1	31.7	21.9
+ feature mapping	11.8	16.0	**14.5**	**27.2**	**20.4**
VTS [45]	6.9	15.1	11.8	23.3	17.8

Using the multi-condition models results in a much better baseline of 23 percent. AFE does not provide any performance improvements in this case. This is expected as the mean-variance normalized MFCC features are fairly robust [28]. Direct feature mapping and the proposed front-end now have similar performance, which in turn is similar to the WER obtained using the clean models with the proposed front-end. Compared to the noisy baseline, the proposed front-end obtains a relative improvement 11.3%. Interestingly, the best performance in noisy conditions is obtained when only T-F masking is used (14.3 percent, a 16.8% relative improvement compared to the baseline).

Although not directly comparable because of modeling differences, we note that the results obtained by the proposed front-end are significantly better than those obtained using MMSE/MMI-SPLICE on this corpus [9]. The ASR models used in [9] are similar to ours. Feature enhancement is performed by learning transformations using the stereo training data; the transformation is constrained to be an offset. The results in [9] show that MMSE SPLICE improves clean and averaged noisy performance from 8.4 to 8.3 and 33.9 to 29.2 percent, respectively, when using clean models. MMI SPLICE, which works better than MMSE SPLICE, improves clean and noisy performance from 14.0 to 13.4 and 19.2 to 18.8 percent, respectively, when using multi-condition models. Results on sets 8 to 14 are not presented in [9]. For the sake of comparison, Table I also shows the performance reported in [45] using VTS based model adaptation. The proposed front-end reduces the gap in performance between model-based and feature-based

TABLE II
WORD ERROR RATES ON THE AURORA-4 CORPUS USING THE DNN-HMM SYSTEMS TRAINED IN CLEAN CONDITIONS. THE BEST PERFORMANCE IN EACH CONDITION IS MARKED IN BOLD

System	Clean	Noisy	Clean + Channel	Noisy + Channel	Average
Cepstral features					
Noisy	5.3	19.2	18.6	36.1	25.4
+ dFDLR	**5.2**	19.6	17.5	35.9	25.4
Feature mapping	5.4	9.9	9.5	22.9	15.1
+ dFDLR	5.4	9.7	8.8	22.3	14.7
Proposed frontend	5.6	9.3	9.5	22.0	14.5
+ dFDLR	5.4	**9.2**	**9.1**	**21.4**	**14.1**
log-mel features					
Noisy	5.2	22.9	21.3	41.6	29.5
+ dFDLR	5.1	22.7	20.5	41.4	29.3
Feature mapping	5.3	10.7	9.6	25.2	16.4
+ dFDLR	5.3	10.3	**8.7**	24.5	15.9
Proposed frontend	5.2	9.5	9.6	24.0	15.4
+ dFDLR	**4.9**	**9.2**	9.0	**23.3**	**14.9**

TABLE III
WORD ERROR RATES ON THE AURORA-4 CORPUS USING THE DNN-HMM SYSTEMS TRAINED ON THE MULTI-CONDITION SET. THE BEST PERFORMANCE IN EACH CONDITION IS MARKED IN BOLD

System	Clean	Noisy	Clean + Channel	Noisy + Channel	Average
Cepstral features					
Noisy	6.7	10.9	10.4	21.9	15.3
+ dFDLR	6.7	11.0	9.8	21.4	15.1
Proposed frontend	6.7	10.0	10.2	20.3	14.2
+ dFLDR	6.8	9.7	9.5	19.7	13.7
Concat-features	5.6	8.9	9.4	20.5	13.6
+ dFDLR	**5.4**	**8.7**	**8.8**	**20.0**	**13.3**
log-mel features					
Noisy	5.3	8.5	9.0	18.2	12.5
+ dFDLR	5.1	8.5	8.4	**17.6**	**12.1**
Proposed frontend	5.3	9.0	8.9	21.9	14.3
+ dFDLR	5.3	8.8	8.7	21.3	13.9
Concat-features	4.9	8.4	8.3	20.4	13.3
+ dFDLR	**4.8**	**8.2**	**8.1**	20.0	13.0

techniques; in noisy conditions masking based front-end even outperforms VTS adaptation.

B. Deep Neural Networks

We first present the results obtained using models trained in clean conditions. The results obtained using both cepstral and log-mel features, with and without dFDLR adaptation, are shown in Table II. Using cepstral features, the baseline performance improves by 22.6% compared to GMMs. This improvement is similar to those reported in other DNN studies. Using dFDLR improves the baseline WER when there is channel mismatch, but the average performance remains unchanged. Using feature mapping alone improves the WER to 15.1 percent. The proposed front-end improves it further to 14.5 percent. dFDLR improves performance in every condition and further reduces the WER to 14.1 percent. Compared to the baseline, this is a 44.5% relative improvement. Similar trends are observed when the models are trained using log-mel features. An average WER of 14.9% is obtained using the proposed front-end with dFDLR adaptation, a 49.5% relative improvement compared to the baseline. It can also be noticed that the models trained using cepstral features work better than those trained using log-mel features.

The performance obtained using the models trained from the multi-condition set are shown in Table III. The table shows results using models trained directly using the noisy features, and those trained using the features obtained after speech separation concatenated with noisy features (Concat-features). For the sake of comparison, we also show the results obtained only using the features after separation (Proposed frontend). As has been noted earlier, this performs worse than Concat-features. When using cepstral models, Concat-features improves performance from 15.3% to 13.6 percent. Using dFDLR improves it further to 13.3 percent, a 13.1% relative improvement compared to the baseline. Contrary to the above trends, when models are trained in the log-mel domain, the noisy baseline performs the best. The average improvement is mainly due to the strong performance on test sets 9 to 14 that have both additive noise and channel mismatch. On those sets, the WER of the noisy baseline is 2 percentage points better than Concat-features, on average. In all other conditions, Concat-features works slightly better. The trend remains unchanged after applying dFDLR. To the best of our knowledge, the average WER of 12.1% obtained by the noisy baseline after dFDLR adaptation is the best published result on Aurora-4. Further, our results in clean, noisy and noisy + channel mismatched conditions (marked in boldface in the table) either match or outperform the best previously published results on these subsets.

IV. DISCUSSION

The best performing systems on Aurora-4 typically use strategies like multiple iterations of model adaptation, speaker specific adaptation using batch updates [45], and discriminative training of GMM-HMM systems [14]. Our systems compare well with such methods even without using the proposed dFDLR feature adaptation. To the best of our knowledge, the only studies that evaluate DNN based acoustic models on this task are [48], [37]. These very recent studies obtain performance close to those reported in the current work using a recently introduced training strategy for the DNNs called dropout training [20]. Although the performance obtained by our system is slightly better, we expect dropout to further improve performance of the systems described in this work.

Several interesting observations can be made from the results presented in the previous section. Firstly, the results clearly show that the speech separation front-end is doing a good job at removing noise and handling channel mismatch. In most cases, when the proposed front-end is used, the performance obtained with clean models is close to those obtained with models trained from the multi-condition data. When using GMMs, the difference in performance is only 0.2 percent. With DNNs this difference is a little larger: 0.7% with cepstral features and 1.9% with log-mel features. Secondly, with no channel mismatch, T-F masking alone worked well in removing noise as can be inferred from the results in Table I. Even though masking improves performance compared to the baseline in the presence of both noise and channel mismatch, its performance is found to be insufficient. In such conditions, feature mapping provides significant gains. Finally, directly performing feature mapping from noisy features to clean features performs reasonably, but it does not perform as well as the proposed front-end. We have also observed in experiments not reported here that directly using the masked features as input to perform feature mapping or acoustic modeling does not perform as well as using them in conjunction with the noisy features. We believe this is because of the distortions introduced by masking, especially in the presence of channel mismatch, and possibly the reduced variability in the training data as noted in [37].

One surprising result from our experiments is that when the DNN models are trained in clean conditions, cepstral features worked better than log-mel features. This trend is reversed when the models are trained using the multi-condition training set. It is likely that in the presence of severe mismatch in training and testing conditions, cepstral features generalize better. Other studies have observed that in matched conditions, log-mel features work better [26], [37].

It has been noted in prior work that DNN-HMM systems work really well when the training and testing conditions do not differ much [48]. We believe that robust feature representations and enhancement techniques will be really helpful if a mismatch in training and testing conditions is anticipated and cannot be handled at the time of training the acoustic models. Background noise can be highly unpredictable and can pose such problems to well trained ASR systems. It should be pointed out that although the speech separation front-end is trained, in this work, using the same dataset as used by the acoustic models, it can also be trained independently using a separate set. Such a training set will not need word-level transcriptions, unlike the data used to train the ASR models.

Generalization to unseen conditions is a perennial issue for supervised learning algorithms, and exists in both acoustic modeling and speech separation. In the case of supervised separation, this problem has been addressed in previous studies and is vigorously pursued currently [23], [43]. Advances on this topic would clearly help improve performance of feature-based systems like the one described in this work.

It is interesting to note that the best average performance of 12.1% is obtained using noisy log-Mel features, with dFDLR adaptation, as input to a DNN based ASR system. Surprisingly, this result also improves upon the previous best result reported on this corpus either using DNNs [37] (which uses only one recognition pass unlike our final system) or GMMs [45]. As in other ASR tasks [5], when the amount of mismatch between training and testing is not significant, DNNs seem to capture sufficient information with good generalization characteristics [48]. Unless a system has ideal knowledge (e.g., the true instantaneous SNR at each T-F unit), any front-end processing is bound to introduce some distortions. In the case of DNNs, such distortions seem to have a detrimental effect on performance [37]. A very recent study on the role of speech enhancement when using DNN based acoustic models makes similar observations [6]. This study shows improvements in performance on a small vocabulary task at much lower SNRs using an enhancement front-end and cepstral features as input to the DNN. It is likely that the improvements will be lower using log-mel features, as shown in our experiments.

This work leads to several interesting research directions. Clearly, better DNN training strategies like dropout and network architectures like maxout [16] are expected to improve performance of all components of the proposed system. From the results in Table III, it can be seen that feature mapping introduces some distortion in the presence of both noise and channel mismatch, resulting in a drop in performance using log-mel features. Future work should focus on developing better feature mapping strategies, e.g., using recurrent neural networks or bidirectional long-short term memories [46]. It will also be interesting to study if alternative feature representations, like the AFE features, are more suitable for feature mapping. The different components of our system are all trained independent of each other. It is possible to treat the whole system, including T-F masking, feature mapping, and acoustic modeling, as a single deep network interconnected through layers with static weights. The static layers correspond to operations like mel filtering, cepstral transformation, calculation of delta components, and mean-variance normalization. With such a formulation, it may be possible to adjust the weights of the entire network, leading to a system that learns to handle the errors made by the layers preceding it. Initial attempts at training the feature mapping function and the acoustic model jointly resulted in improved frame classification performance, but it did not translate to improved WER. One could potentially use posterior modeling [31] to translate the frame level improvements into ASR performance. Future work will explore this idea further. Finally, it may be worth exploring alternative ways of using the output of speech separation when using DNN based acoustic models. For example, [37] proposed passing a crude noise estimate as an additional feature to the DNNs. A more accurate noise estimate from a separation front-end would help improve performance of such a system.

To conclude, we have proposed a speech separation front-end based on T-F masking and feature mapping that significantly improves ASR performance. A semi-supervised feature adaptation technique called dFDLR is proposed which can be applied on a per-utterance basis. The final system produces state-of-the-art performance on the Aurora-4 medium-large vocabulary recognition task. The results show that supervised feature-based ASR techniques have considerable potential in improving performance.

Acknowledgment

The authors would like to thank Yuxuan Wang for useful discussions and Dong Yu for providing details about DNN training.

References

[1] R. Astudillo and R. Orglmeister, "Computing MMSE estimates and residual uncertainty directly in the feature domain of ASR using STFT domain speech distortion models," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 21, no. 5, pp. 1023–1034, May 2013.

[2] L. Bottou and O. Bousquet, "The tradeoffs of large-scale learning," *Adv. Neural Inf. Process. Syst. 20*, pp. 161–168, 2008.

[3] C.-P. Chen and J. A. Bilmes, "MVA processing of speech features," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 15, no. 1, pp. 257–270, Jan. 2007.

[4] G. E. Dahl, T. N. Sainath, and G. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *Proc. IEEE ICASSP*, 2013, pp. 8609–8613.

[5] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 20, no. 1, pp. 30–42, Mar. 2012.

[6] M. Delcroix, Y. Kubo, T. Nakatani, and A. Nakamura, "Is speech enhancement pre-processing still relevant when using deep neural networks for acoustic modeling?," in *Proc. Interspeech*, 2013, pp. 2992–2996.

[7] L. Deng, A. Acero, L. Jiang, J. Droppo, and X. Huang, "High-performance robust speech recognition using stereo training data," in *Proc. IEEE ICASSP*, 2001, pp. 301–304.

[8] L. Deng, J. Droppo, and A. Acero, "Dynamic compensation of HMM variances using the feature enhancement uncertainty computed from a parametric model of speech distortion," *IEEE Trans. Speech Audio Process.*, vol. 13, no. 3, pp. 412–421, May 2005.

[9] J. Droppo, "Feature compensation," in *Techniques for Noise Robustness in Automatic Speech Recognition*, T. Virtanen, B. Raj, and R. Singh, Eds. West Sussex, U.K.: Wiley, 2012, ch. 9, pp. 229–250.

[10] J. Du, Y. Hu, L.-R. Dai, and R.-H. Wang, "HMM-based pseudo-clean speech synthesis for splice algorithm," in *Proc. IEEE ICASSP*, 2010, pp. 4570–4573.

[11] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, 2010.

[12] D. P. W. Ellis, "PLP and RASTA (and MFCC, and inversion) in Matlab," 2005 [Online]. Available: http://www.ee.columbia.edu/dpwe/resources/matlab/rastamat/

[13] *Speech processing transmission and quality aspects (STQ); Distributed speech recognition; Advanced front-end feature extraction algorithm; Compression algorithms*, ES 202 050 V1.1.4, ETSI, 2005.

[14] F. Flego and M. J. F. Gales, "Discriminative adaptive training with VTS and JUD," in *Proc. IEEE ASRU*, 2009, pp. 170–175.

[15] M. J. F. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Comput. Speech Lang.*, vol. 12, no. 2, pp. 75–98, 1998.

[16] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *J. Mach. Learn. Res. Workshop Conf. Proc.*, vol. 28, no. 3, pp. 1319–1327, 2013.

[17] K. Han and D. L. Wang, "A classification based approach to speech segregation," *J. Acoust. Soc. Amer.*, vol. 132, no. 5, pp. 3475–3483, 2012.

[18] W. Hartmann, A. Narayanan, E. Fosler-Lussier, and D. L. Wang, "A direct masking approach to robust ASR," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 21, no. 10, pp. 1993–2005, Oct. 2013.

[19] H. Hermansky and N. Morgan, "RASTA processing of speech," *IEEE Trans. Speech Audio Process.*, vol. 2, no. 4, pp. 578–589, Oct. 1994.

[20] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[21] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[22] G. Kim, Y. Lu, Y. Hu, and P. Loizou, "An algorithm that improves speech intelligibility in noise for normal-hearing listeners," *J. Acoust. Soc. Amer.*, vol. 126, no. 3, pp. 1486–1494, 2009.

[23] W. Kim and R. Stern, "Mask classifcation for missing-feature reconstruction for robust speech recognition in unknown background noise," *Speech Commun.*, vol. 53, pp. 1–11, 2011.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 25, pp. 1106–1114, 2012.

[25] A. L. Maas, Q. V. Le, T. M. O'Neil, O. Vinyals, P. Nguyen, and A. Y. Ng, "Recurrent neural networks for noise reduction in robust ASR," in *Proc. Interspeech*, 2012.

[26] A. Mohamed, G. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 20, no. 1, pp. 14–22, Jan. 2012.

[27] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. ICML 27*, 2010, pp. 807–814.

[28] A. Narayanan and D. Wang, "Coupling binary masking and robust ASR," in *Proc. IEEE ICASSP*, 2013, pp. 6817–6821.

[29] A. Narayanan and D. Wang, "Ideal ratio mask estimation using deep neural networks for robust speech recognition," in *Proc. IEEE ICASSP*, 2013, pp. 7092–7096.

[30] N. Parihar and J. Picone, "Analysis of the Aurora large vocabulary evaluations," in *Proc. Eurospeech*, 2003, pp. 337–340.

[31] R. Prabhavalkar, T. N. Sainath, D. Nahamoo, B. Ramabhadran, and D. Kanevsky, "An evaluation of posterior modeling techniques for phonetic recognition," in *Proc. IEEE ICASSP*, 2013, pp. 7165–7169.

[32] B. Raj and R. Stern, "Missing-feature approaches in speech recognition," *IEEE Signal Process. Mag.*, vol. 22, no. 5, pp. 101–116, 2005.

[33] N. Roman, D. L. Wang, and G. J. Brown, "Speech segregation based on sound localization," *J. Acoust. Soc. Amer.*, vol. 114, no. 4, pp. 2236–2252, 2003.

[34] N. Roman and J. Woodruff, "Intelligibility of reverberant noisy speech with ideal binary masking," *J. Acoust. Soc. Amer.*, vol. 130, no. 4, pp. 2153–2161, 2011.

[35] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. IEEE ASRU*, 2011, pp. 24–29.

[36] M. L. Seltzer, B. Raj, and R. M. Stern, "A Bayesian classifier for spectrographic mask estimation for missing feature speech recognition," *Speech Commun.*, vol. 43, no. 4, pp. 379–393, 2004.

[37] M. L. Seltzer, D. Yu, and Y.-Q. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Proc. IEEE ICASSP*, 2013, pp. 7398–7402.

[38] S. Srinivasan, N. Roman, and D. L. Wang, "Binary and ratio time-frequency masks for robust speech recognition," *Speech Commun.*, vol. 48, pp. 1486–1501, 2006.

[39] K. Vertanen, "HTK Wall Street Journal training recipe," 2005 [Online]. Available: http://www.keithv.com/software/htk/

[40] D. L. Wang, "On ideal binary masks as the computational goal of auditory scene analysis," in *Speech Separation by Humans and Machines*, P. Divenyi, Ed. Boston, MA, USA: Kluwer, 2005, pp. 181–197.

[41] D. L. Wang, U. Kjems, M. S. Pedersen, J. B. Boldt, and T. Lunner, "Speech intelligibility in background noise with ideal binary time-frequency masking," *J. Acoust. Soc. Amer.*, vol. 125, pp. 2336–2347, 2009.

[42] Y. Wang, K. Han, and D. L. Wang, "Exploring monaural features for classification-based speech segregation," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 21, pp. 270–279, 2013.

[43] Y. Wang and D. L. Wang, "Towards scaling up classification-based speech separation," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 21, no. 7, pp. 1381–1390, Jul. 2013.

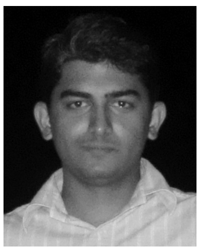
[44] Y. Wang and D. L. Wang, "Feature denoising for speech separation in unknown noisy environments," in *Proc. IEEE ICASSP*, 2013, pp. 7472–7476.

[45] Y.-Q. Wang and M. J. F. Gales, "Speaker and noise factorization for robust speech recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 20, no. 7, pp. 2149–2158, 2012.

[46] F. Weninger, J. Geiger, M. Wöllmer, B. Schuller, and G. Rigoll, "The Munich feature enhancement approach to the 2nd CHiME challenge using BLSTM recurrent neural networks," in *Proc. 2nd CHiME Mach. Listen. Multisource Environ. Workshop*, 2013, pp. 86–90.

[47] S. Young, G. Evermann, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland*, The HTK Book*. Cambridge, U.K.: Cambridge Univ. Press, 2002 [Online]. Available: http://htk.eng.cam.ac.uk.

[48] D. Yu, M. L. Seltzer, J. Li, J.-T. Huang, and F. Seide, "Feature learning in deep neural networks - studies on speech recognition tasks," in *Proc. ICLR*, 2013.



Arun Narayanan (S'11) received the B.Tech. degree in computer science from the University of Kerala, Trivandrum, India, in 2005, and the M.S. degree in computer science from the Ohio State University, Columbus, USA, in 2012, where he is currently pursuing the Ph.D. degree. His research interests include robust automatic speech recognition, speech separation, and machine learning.

DeLiang Wang photograph and biography not provided at the time of publication.