LOW-RESOURCE NOISE-ROBUST FEATURE POST-PROCESSING ON AURORA 2.0

Chia-Ping Chen   *Jeff Bilmes*   *Katrin Kirchhoff*

SSLI Lab
Department of Electrical Engineering
University of Washington
Seattle, WA 98195-2500
{chiaping,bilmes,katrin}@ee.washington.edu

ABSTRACT

We present a highly effective and extremely simple noise-robust front end based on novel post-processing of standard MFCC features. It performs remarkably well on the Aurora 2.0 noisy-digits database without requiring any increase in model complexity. Compared to the Aurora 2.0 baseline system, our technique improves the average word error rate by 45% in the multi-condition training case, (matched training/testing conditions) and 60% in the clean training case (mismatched training/testing conditions) — this is an improvement that rivals some of the best known results on this database. Our method, moreover, improves the performances in all cases, regardless of clean or noisy speech, matched or mis-matched environments. Our technique is entirely general because it makes no assumptions about the existence, type, or level of noise in the speech signal. Moreover, its simplicity means that it should be easy to integrate with other techniques in order to yield further improvements.

1. INTRODUCTION

Noise-robustness is today one of the most challenging and important problems in automatic speech recognition (ASR). The performance of ASR systems often decreases dramatically when the noise level increases. Often the degradation is minor when the signal-to-noise ratio (SNR) is high, but quite significant at low SNR levels. In general, there are two types of noise, additive and convolutional. These different noise type extremes corrupt the speech signal in very different ways, so it is therefore challenging to design a methodology that is robust to both. Even given a technique that ideally handles both types of noise, the problem of *mismatch* remains. That is, when the type of test conditions are different from the training conditions (different additive and/or convolutional noises in the training and testing environments), ASR performance is often extremely poor. The severe degradation in the noisy and/or mismatched environments is one of the major obstacles for the realization of ASR systems in the full variety of adverse acoustic environments that humans are prone to encounter.

The Aurora 2.0 database [1] provides an excellent platform in which to research noise-robustness techniques. In the past, a variety of sophisticated techniques have been applied to this data set. For example, in [2], a front end consisting of principle component analysis and a discriminative neural network applied to two types of speech features, and a backend consisting of standard Gaussian mixture acoustic models is used. In [3], missing-data theory is used by identifying reliable features in the spectral-temporal domain. In [4], an algorithm for signal estimation in the cepstral domain is implemented. In [5], voice activity detector and variable frame rate techniques are used to drop noisy feature vectors to reduce the insertion errors. In [6], nonlinear spectral subtraction, noise masking, feature filters, and model adaptation techniques are used. In [7], data-driven temporal filters, on-line mean and variance normalization, voice activity detection, and server side discriminant features are integrated together to improve noise-robustness. Overall, each of the cases above yield significant performance improvements on Aurora 2.0.

In this paper, we propose a feature post-processing technique that is both extremely simple and also remarkably effective for noise-robustness on Aurora 2.0. Our method was discovered by visually inspecting the cepstral-domain time sequences of the same utterance in both the clean and the noisy environments (figures are included later in the paper). Our experiments show that this approach does achieve a performance level that is comparable to the currently best known technique [2][1].

This paper is organized as follows. In section 2, we describe our feature post-processing technique. In section 3, we analyze our proposed method mathematically. In section 4, we present experimental results and compare them with results from the baseline system defined in [1]. In section 5, we draw conclusions and describe future research.

2. DESCRIPTION

In this section, we describe our feature post-processing methodology. Note that the post-processing described below will at first appear quite similar to certain schemes well known to the community (namely variance normalization and mean subtraction). The crucial difference between this and past work, however, lies in the domain in which the post-processing is applied. We start with standard mel-frequency cepstral coefficients (MFCC), $c_0 \dots c_{12}$, along with their deltas and double-deltas as our raw features. For a given utterance, we represent the data by a matrix C whose element C_{td} is the dth component of the feature vector at time t, $t = 1 \dots T$, the number of frames in the utterance and $d = 1 \dots D$, the dimension of the feature space. In other words, each row of C represents a feature vector and each column represents a time sequence. The first step is standard mean subtraction (MS) defined by:

$$C'_{td} = C_{td} - \mu_d \tag{1}$$

where

$$\mu_d = \frac{1}{T}\sum_{t=1}^{T} C_{td} \tag{2}$$

This is followed by the variance normalization (VN) defined by:

$$\bar{C}_{td} = \frac{C'_{td}}{\sigma_d} = \frac{C_{td} - \mu_d}{\sigma_d} \tag{3}$$

[1]On the Aurora 2.0 multi-condition training set.

SNR/dB	test A	test B	test C	average	baseline[1]
clean	98.80	98.80	98.51	98.74	98.52
20	98.89	98.80	98.67	98.81	97.35
15	98.26	98.29	98.05	98.23	96.29
10	96.71	96.72	96.84	96.74	93.78
5	91.80	91.90	91.44	91.77	85.51
0	77.65	76.63	77.32	77.18	58.99
-5	47.08	44.78	47.62	46.27	24.49
avg 0-20	92.66	92.47	92.47	92.55	86.39

SNR/dB	test A	test B	test C	average	baseline[1]
clean	99.18	99.18	99.04	99.15	99.03
20	97.72	97.98	97.55	97.79	94.07
15	95.73	96.39	95.61	95.97	85.03
10	90.88	92.33	90.78	91.44	65.51
5	80.10	81.91	80.31	80.87	38.60
0	57.80	60.54	57.25	58.79	17.09
-5	28.50	29.26	28.55	28.81	8.53
avg 0-20	84.44	85.83	84.30	84.97	60.06

Table 1. Word accuracies (as percentages) for our methodology. The total number of free parameters is 42k. Top: multi-condition training (i.e., matched training/testing conditions); bottom: clean training (i.e., mis-matched training/testing conditions).

order	clean test	noisy test	noisiest test
$M = 1$	98.32	92.27	43.99
$\boldsymbol{M = 2}$	**98.74**	**92.55**	**46.27**
$M = 3$	98.64	92.26	45.64
$M = 4$	98.29	92.09	45.85
$M = 10$	96.79	87.47	37.32

order	clean test	noisy test	noisiest test
$M = 1$	99.41	81.69	22.42
$M = 2$	99.21	84.24	28.21
$M = 3$	99.19	83.99	28.35
$\boldsymbol{M = 4}$	**99.15**	**84.97**	**28.81**
$M = 10$	97.56	80.52	27.79

Table 2. Comparison of different order ARMA filters. Word accuracies for clean test speech, the noisy test speech (average over SNR= $20,\ 15,\ 10,\ 5,\ 0$ dB) and the noisiest test speech (SNR= -5dB). Top: multi-condition training; bottom: clean training.

	train condition		word acc.	WER im-
	multi	clean	average	provement
baseline[1]	86.39	60.06	73.23	=
MS+VN ($M = 0$)	91.40	78.25	84.83	43%
MS+VN+ARMA	92.55	84.97	88.76	58%

Table 3. Step-wise improvement of our technique. Word accuracies in noisy (SNR= $0,\ 5,\ 10,\ 15,\ 20$dB) test speech ($M = 2$ for multi-condition training and $M = 4$ for clean training). Note that relative improvements (on the right) are word error rate (WER) improvements.

	test A	test B	test C	average
non-causal ARMA	92.66	92.47	92.47	92.55
non-causal MA	92.36	92.42	92.12	92.34
causal ARMA	92.43	92.62	92.03	92.43
causal MA	92.08	92.38	91.92	92.17

	test A	test B	test C	average
non-causal ARMA	83.79	85.05	83.54	84.24
non-causal MA	84.02	85.75	84.85	84.88
causal ARMA	84.19	85.32	83.94	84.59
causal MA	81.45	82.71	81.23	81.91

Table 4. Comparison of different low-pass filters. All the filters have $M = 2$. Shown in the table are word accuracies for *noisy* test speech. Top: multi-condition training; bottom: clean training.

$$\begin{tikzcd} wave \arrow[r] & \boxed{\text{FE}} \arrow[r, dash, "C"] & \arrow[r] & \boxed{\text{MS+VN}} \arrow[r, dash, "\bar{C}"] & \arrow[r] & \boxed{\text{ARMA}} \arrow[r, dash, "\breve{C}"] & \arrow[r] & \boxed{\text{HMM}} \end{tikzcd}$$

Fig. 1. Block diagram of our feature post-processing technique. Abbreviations are : FE – feature extraction; MS – mean subtraction; VN – variance normalization; ARMA – mixed auto-regression and moving average filter; and HMM – hidden Markov model, the standard backend for training and decoding.

where

$$\sigma_d = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(C_{td} - \mu_d)^2} \tag{4}$$

The third step is processing by a mixed auto-regression moving average (ARMA) filter, defined by:

$$\breve{C}_{td} = \begin{cases} \frac{\sum_{i=1}^{M}\breve{C}_{(t-i)d} + \sum_{j=0}^{M}\bar{C}_{(t+j)d}}{2M+1} & \text{if } M < t \leq T - M, \\ \bar{C}_{td} & \text{otherwise} \end{cases} \tag{5}$$

where M is the order of the ARMA filter. The special case of $M = 0$ degenerates to no ARMA filtering.

A block diagram of these post-processing techniques is provided in Figure 1. The processing is performed individually for each Aurora 2.0 training and testing utterance. Note that we perform the variance normalization and the ARMA filtering in the *same* domain, namely both in the cepstral domain. This is different from previously proposed approaches that perform filtering (often in the spectral or log-spectral domain) and the variance normalization (often in the cepstral domain) in *different* domains [8]. Also note that the filter is the same for each time sequence and is not trained from specific data. This means that the filter is not dependent or tuned to a particular type of noise.

3. ANALYSIS

In equation (5), each mean-subtracted and variance-normalized time sequence is further processed by an ARMA filter. The ARMA filter used is essentially a low-pass filter, smoothing out any spikes in the time sequence. As will be seen, this filtering operation results in further significant improvements. The idea of smoothing out a spiky time sequence is quite natural. While in clean speech the spikes might contain important information about the speech utterance, in noisy speech these spikes are more likely to be caused by noise. Therefore, there is an inherent trade-off in choosing the order M of the filter. A small M will retain the short-term cepstral information but is more vulnerable to noise, while a large M will make the processed features less corrupted by noise, but the short-term cepstral information will be lost. Intuitively, the most extreme cases of $M = 0$ or $M \gg 1$ would have the poorest performance when the speech is noisy. This suggests that the optimal M^* will be a small positive integer, something that we experimentally verify later in the paper.

An example is illustrated in Figure 2, which shows one of the cepstral coefficients (c_1) over time of the same digit-string uttered under different SNRs. Note how the mean subtraction and variance normalization combine to bring the time sequences in different noise levels to the same relative level (via mean subtraction) and scale (via variance normalization). The low-pass ARMA filter smooths out the sequences toward temporal similitude, further minimizing the differences between the clean and the noisy plots. While there is always a danger in using visual inspection to deduce a processing methodology for speech recognition, we have found that these simple processing steps, when applied in the cepstral domain, have a remarkably positive influence on word error as will be seen below.

In order to examine the relationship between $\bar{C}$ and $\breve{C}$ in the frequency domain, we can rewrite equation (5) as:

$$(2M+1)\breve{C}_{td} - \breve{C}_{(t-1)d} - \cdots - \breve{C}_{(t-M)d} = \bar{C}_{td} + \cdots + \bar{C}_{(t+M)d} \tag{6}$$

From (6), the transfer function is:

$$H(z) = \frac{1 + z + \cdots + z^M}{2M + 1 - z^{-1} - \cdots - z^{-M}} \tag{7}$$

The frequency response of the ARMA filter of order M is:

$$\begin{aligned} H(e^{j\omega}) &= \frac{1 + e^{j\omega} + \cdots + e^{jM\omega}}{2M + 1 - e^{-j\omega} - \cdots - e^{-jM\omega}} \\ &= \frac{1 - e^{j(M+1)\omega}}{2M + 2 - (2M+1)e^{j\omega} - e^{-jM\omega}} \end{aligned} \tag{8}$$

Note that for $\omega = 0$, $H(e^{j\omega}) = 1$. There are $\lfloor \frac{M+1}{2} \rfloor$ zeros in the interval $[0, \pi]$ equally spaced at

$$\omega = 2n\pi/(M+1), n = 1, 2, \ldots, \lfloor \frac{M+1}{2} \rfloor \tag{9}$$

The equation (8) (9) state that the number of zeros in the frequency response of the ARMA filter is approximately proportional to its order. They support the intuition that a large M will perform poorly since it could filter out important speech information. The frequency responses of the cases $M = 2, 4$ are plotted in Figure 3.

4. EVALUATION

We evaluate our methodology on the Aurora 2.0 noisy digits database. On this database, two training sets and three test sets are defined [1]. The multi-condition training set consists of both clean and noisy speech (so this constitutes matched training/testing conditions), while the clean training set consists only of clean speech (constituting much more mismatched training/testing conditions). Test set A is composed of speech with conditions matched to the multi-condition training set, test set B is composed of speech with non-matched background noise, and test set C is composed of speech with partly matched background noise and non-matched convolutional noise.

In the experiments, we use a simple HMM-based system using whole-word models, 16 states per word, 3 Gaussian components per state, uniform segmental k-means Gaussian initialization, and an EM-training allowing a Gaussian component to vanish when the component weight gets small. During training, we use a 3-state silence model at the beginning and the end of each utterance, and a forced single-state short-pause model between words. During testing, we allow the silence model to optionally occur between words. In all experiments, the total number of model parameters is no more than 42k, which is virtually the same as the number in the baseline system.

We summarize the recognition results on the clean test speech, the noisy test speech (SNR= $20, 15, 10, 5, 0$ dB) and the noisiest test speech (SNR= -5dB) in Table 1. In both cases of the clean and multi-condition training, our post-processing technique results in significant improvements in all types of test conditions over the baseline results. Note that it's not unusual for a noise-robust technique to degrade the clean-case performance, but our technique improves over the baseline results in all test conditions.

A comparison of different orders of the ARMA filtering is given in Table 2. We observe that the optimal value of M differs in the multi-condition training and clean training cases. In the multi-condition training condition, $M = 2$ yields the best results (shown in boldface). This value apparently strikes a good balance between information preservation and noise robustness. In clean training, $M = 4$ (also shown in boldface) yields the best results. Larger M corresponds to longer window. Apparently, it is necessary to use a wider temporal window to produce more reliable features when the conditions are mismatched.

We also test the effectiveness of our technique both with and without the ARMA filtering to better understand its relative merits. The results of these experiments are summarized in Table 3. Note that the mean subtraction is applied in both cases. The results show that while variance normalization and mean subtraction improves performance over the baseline, the addition of the ARMA filter provides significant further improvements.

The ARMA filter that we used in (5) is a non-causal low-pass filter. In order to see whether there is an intrinsic advantage to this particular ARMA filtering method, or if any low-pass filter suffices, we ran the same experiments with the ARMA filter replaced by the following low-pass filters.

- causal ARMA filter

$$\breve{C}_{td} = \begin{cases} \frac{\sum_{i=1}^{M} \breve{C}_{(t-i)d} + \sum_{j=0}^{M} \bar{C}_{(t-j)d}}{2M+1} & \text{if } M < t \le T, \\ \bar{C}_{td} & \text{otherwise} \end{cases}$$

- non-causal MA filter

$$\breve{C}_{td} = \begin{cases} \frac{\sum_{i=-M}^{M} \bar{C}_{(t+i)d}}{2M+1} & \text{if } M < t \le T - M, \\ \bar{C}_{td} & \text{otherwise} \end{cases}$$

- causal MA filter

$$\breve{C}_{td} = \begin{cases} \frac{\sum_{i=0}^{M} \bar{C}_{(t-i)d}}{M+1} & \text{if } M < t \le T, \\ \bar{C}_{td} & \text{otherwise} \end{cases}$$

A comparison of our experimental results where we replace the non-causal ARMA filter with these filters is given in Table 4. Note that because of the way we defined the causal MA filter, the results are given for a 3- (rather than 5-) tap filter in that case. As can be seen, the difference in performances between causal/non-causal and ARMA/MA filters is small. These results lead us to the conclusion that it is sufficient to use a general low-pass filter in the cepstral domain to achieve good performance for this task. The non-causal ARMA filter, however, has an extremely simple implementation since the array computation can be done entirely in-place.

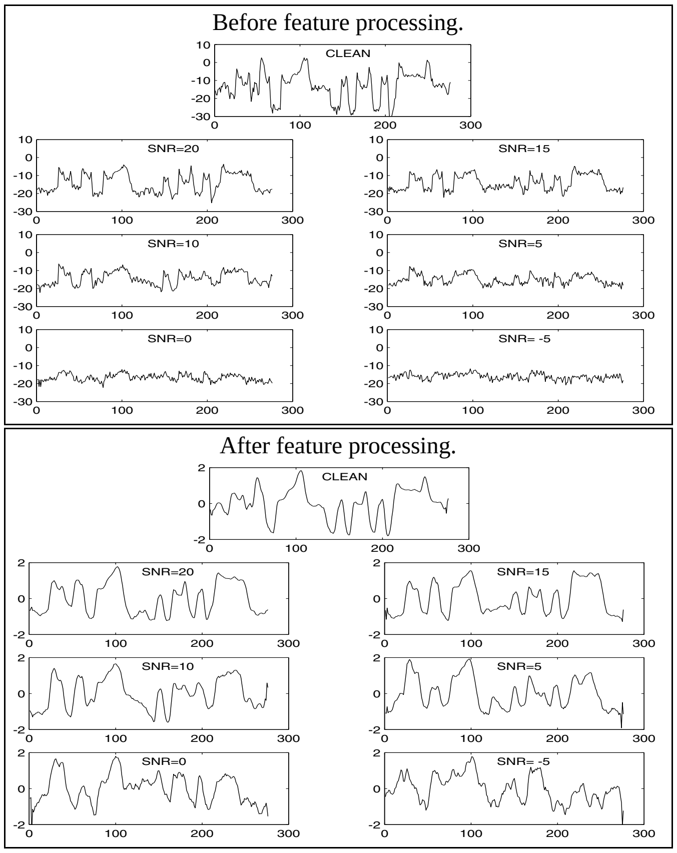

Fig. 2. The time sequences of the cepstral coefficient c_1 for the digit string 5376869 corrupted with different levels of noises, both before and after our post-processing. In this case, $M = 3$.

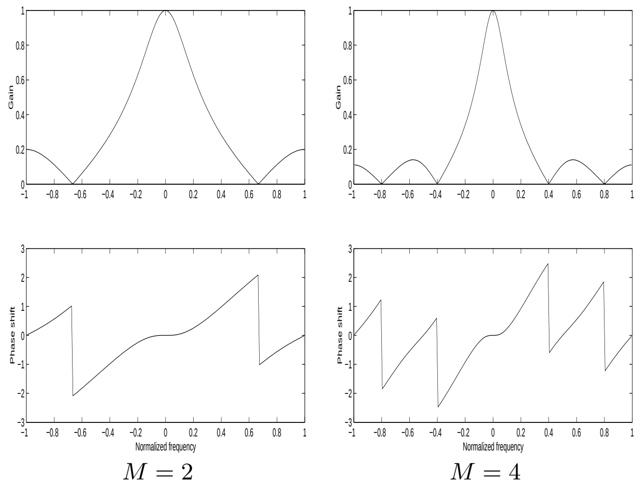

Fig. 3. Gains and phase shifts of the ARMA filters. Here 1Hz in normalized frequency corresponds to 50Hz in the cepstral modulation domain.

5. SUMMARY AND FUTURE WORK

In this paper, we propose a feature post-processing methodology which is extremely simple yet extremely effective on the Aurora 2.0 noisy digits database. It achieves the same level of performance as those systems which use much more sophisticated acoustic modeling and/or speech enhancements techniques. In general, our improvements are comparable to systems with many more free parameters, and which have significantly greater computational and model complexity. Our method does not require any knowledge about the noise types, and performs well in both the matched and mis-matched training and testing environments. Furthermore, the technique requires no additional model parameters and does not require any computationally expensive training algorithms for parameter learning.

The experimental results with the clean training set and the noisy test set are of particular interest because of the implication to robustness in mis-matched acoustic environments. A technique where it is possible to train only in clean speech and have good results in both clean and noisy speech is certainly desirable.

Since our feature post-processing scheme is performed within the feature space, there are no difficulties to combine our technique with other noise-robust techniques. Further, we plan to experiment with extensions to our method, to post-process other types of feature streams, and to combine it with more sophisticated acoustic models.

6. REFERENCES

[1] H. G. Hirsch and D. Pearce, "The AURORA Experimental Framework for the Performance Evaluations of Speech Recognition Systems under Noisy Conditions", ISCA ITRW ASR2000, Paris, September 2000.

[2] D. Ellis and M. Gomez, "Investigations into Tandem Acoustic Modeling for the Aurora Task", pp. 189-192, Proceedings Eurospeech 2001.

[3] J. Barker, M. Cooke, and P. Green, "Robust ASR Based on Clean Speech Models: An Evaluation of Missing Data Techniques for Connected Digit Recognition in Noise", pp. 213-216, Proceedings Eurospeech 2001.

[4] J. Droppo, L. Deng and A. Acero, "Evaluation of the SPLICE Algorithm on the Aurora2 Database", pp. 217-220, Proceedings Eurospeech 2001.

[5] Johan de Veth, Laurent Mauuary, Bernhard Noe, Febe de Wet, Juergen Sienel, Louis Boves and Denis Jouver, "Feature Vector Selection to Improve ASR Robustness in Noisy Conditions", pp. 201-204, Proceedings Eurospeech 2001.

[6] M. Lieb and A. Fischer, "Experiments with the Philips Continuous ASR System on the Aurora Noisy Digits Database", pp. 625-628, Proceedings Eurospeech 2001.

[7] C. Benitez, L. Burget, B. Chen, S. Dupont, H. Garudadri, H. Hermansky, P. Jain, S. Kajarekar, N. Morgan, S. Sivadas, "Robust ASR Front-end Using Spectral-based and Discriminant Features: Experiments on the Aurora Tasks", pp. 429-432, Proceedings Eurospeech 2001.

[8] H. Hermansky, N. Morgan, "RASTA Processing of Speech", IEEE Transactions on Speech and Audio Processing, Vol. 2, No. 4, pp. 578-589, Oct. 1994.