MVA Processing of Speech Features

Chia-Ping Chen, Jeff Bilmes
{chiaping,bilmes}@ee.washington.edu

SSLI Lab
Dept of EE, University of Washington
Seattle, WA 98195-2500
MVA Processing of Speech Features

Chia-Ping Chen, Jeff Bilmes
{chiaping,bilmes}@ee.washington.edu

SSLI Lab
Dept of EE, University of Washington
Seattle, WA 98195-2500

University of Washington, Dept. of EE, UWEETR-2003-0024
November 2003

Abstract

In this paper, we investigate a technique consisting of mean subtraction, variance normalization and time sequence filtering. Unlike other techniques, it applies auto-regression moving-average (ARMA) filtering to the time sequence in the cepstral domain. We called this technique the MVA post-processing and the speech features with MVA post-processing the MVA features. Overall, compared to raw features without MVA post-processing, MVA features achieve improvements of 45% on matched tasks and 65% on mismatched tasks on the Aurora 2.0 noisy speech database, and well above a 50% improvement on the Aurora 3.0 database. These improvements are comparable to systems with much more complicated techniques even though MVA is relatively simple and requires practically no additional computational cost. In this paper, in addition to describing MVA processing, we also present a novel analysis of the distortion of mel-frequency cepstral coefficients and the log energy in the presence of different types of noises. The effectiveness of MVA is extensively investigated with respect to several variations: the configurations used to extract raw features, the domains where MVA is applied, the filters that are used, and the orders of the ARMA filters. Specifically, it is argued and demonstrated that MVA works better when applied to the zeroth cepstral coefficient than to the log energy, that MVA works better in the cepstral domain, that an ARMA filter is better than either designed FIR filters or data-driven filters, and that a five-tap ARMA filter is sufficient to achieve good performances in a variety of settings. We also investigate a multi-domain generalization of MVA technique and evaluate its performance. 1

1 Introduction and Overview

Simply put, an automatic speech recognition (ASR) system outputs the most likely hypothetical sentence $W^*$, given the acoustic evidence $A$

$$W^* = \arg \max_W P(A|W)P(W),$$

where $P(A|W)$ is the acoustic model and $P(W)$ is the language model. If these models are accurate, and an admissible (i.e. free of search errors) search algorithm exists, then the ASR problem can in principle be solved in the minimum sentence-error-rate sense.

A main issue here is that the acoustic model (as well as the language model, not discussed in this paper) learned from the training data is only an approximation in test data environments. Such acoustic model inaccuracies for test data has to do with three somewhat separate issues. First, there may be inaccuracy in the assumed parametric form for the acoustic model. The ubiquitous hidden Markov models (HMMs) with state-dependent Gaussian (mixture) density functions belong to this category. Even asymptotically, i.e., in the limit of infinite amount of training data, the inaccuracy of an assumed model (with fixed structure and number of parameters) leads to a non-zero KL-divergence [12] between the learned probability distribution and the true distribution. An account of the capacity/limitation of hidden Markov models is given in [4] and a general framework of segment modeling is given in [37]. Second, the issue of data sparsity, leading to the result that the learned model is different from the asymptotic model, further compromises

1 submitted to IEEE Transactions on Speech and Audio Processing
the accuracy of the acoustic model. Although one can tie parameters for robust estimation [36, 45] to alleviate this problem, the learned model is still not perfect. Lastly, the issue of mismatch makes the acoustic model of the test data different from that of the training data and therefore could altogether invalidate any model learned solely from training data.

Noise-robustness in ASR is crucial since in many applications, a trained ASR system often needs to carry out recognition in a variety of everyday acoustic environments. In dealing with noise-robustness, a major issue is the aforementioned acoustic mismatch, that is, the acoustic models obtained using only training data do not match the test data statistics. Noise in testing environments is one of the key sources that cause such mismatch by causing corruption in speech samples and thus distortion in speech features. By its very nature, purely random noise is difficult to model, and can via its affect on speech features cause spurious likelihood scores leading to recognition errors.

Understanding the systematic corruption and distortion of the speech signal has been the focus of noise-robust ASR research. In the case of corruption, techniques such as Kalman filters [38] and microphone arrays [43] are often used to recover the clean speech samples. In the case of distortion, sets of noise-robust features have been proposed, or alternatively speech enhancement techniques are used to restore the clean speech features. The former includes such features as RASTA-PLP [24], visual features [39] and cross-correlation features [6], while the latter includes such techniques as spectral subtraction [8], cepstral mean normalization [1, 18] and norm equalization [28]. The distortion in speech features can be translated into deformation of the acoustic space, to which the models can be accordingly adjusted. Techniques stemming from such considerations to bridge the gap between training and testing acoustics are often termed model compensation techniques for noise. Such techniques include stochastic matching [40], parallel model combination [44, 21], and model adaptations [20]. Still other ways to achieve noise-robustness can be generally called combination of multiple (complementary) information sources, where multiple feature streams are incorporated in the ASR systems. These different features range from spectral-based features such as mel-frequency cepstral coefficients (MFCC) [13], temporal-based features such as temporal patterns (TRAPS) [25], linear prediction coefficients (LPC) [34], and visual features. Such combination can be done in feature extraction [39], lattice re-scoring [16], or output generation [17, 32]. On small-vocabulary ASR tasks, feature combination (via discriminatively trained neural networks) has been investigated [15], while on medium- to large-vocabulary ASR tasks, hypotheses combination is found to be effective [19, 41, 30].

Even for very small vocabulary ASR tasks, relatively complicated systems have been attempted to achieve good performance. For example, in [15], a front end consisting of principle component analysis and a discriminative neural network applied to two types of speech features, and a back end consisting of standard Gaussian mixture acoustic models is used. In [2], missing-data theory is used by identifying reliable features in the spectral-temporal domain. In [14], voice activity detector and variable frame rate techniques are used to drop noisy feature vectors to reduce the insertion errors. Such systems can be computationally intensive as they often introduce additional parameters either in the feature extraction or in the speech models, and these parameters need to be learned from a given data sample. A fundamental issue here is the risk of over-training [7], which occurs when many spurious parameters exist in the model. Fine tuning these parameters to match a particular set of data may result in performance degradation when used with unseen data.

This paper presents a noise-robust technique that is simple and effective. The technique post-processes speech features. It is implemented in the front end, and belongs in the category of noise-robust features as mentioned above. The fact that this additional signal processing does not assume any additional parameters that must be learned from data provides two advantages: (1) the processing is arguably more robust than data-driven techniques when environmental mismatch is unavoidable, and (2) the computational burden is small which is vital when computational resources are limited.

Note that in earlier work, we have published initial purely empirical results [10, 11]. In this work, we present theoretical justifications along with new improved experimental findings. In particular, we include a novel analysis on the distortion of raw features under typical noise conditions and show that our technique can counter such distortion. Furthermore, we show theoretically and verify experimentally that the effectiveness of this technique strongly depends on the type of raw feature. We also compare results of this technique to a number of variants to show its advantage in performance. Finally, we generalize this technique to multiple domains and various filter orders.

The organization of this paper is as follows. In Section 2, after distinguishing different forms of noise via their effects on speech and raw speech features, we specify our novel technique, MVA post-processing. This is followed by a novel analysis of the distortion of features in Section 3. The experimental speech databases and ASR systems are described in Section 4. Results are presented and analyzed in Section 5. In Section 6, conclusions are summarized.
2 Definitions of Noises, Features and MVA Post-Processing

The noise-robustness of post-processed speech features depends on the noise, the feature set, and the post-processing scheme. In this section, we first define the types of noises which are common in everyday acoustic environments and which have detrimental effects on speech recognition. This is followed by definitions of the raw speech features used in this paper. Finally we specify our post-processing technique. While some of the following definitions are not new in and of themselves, repeating them here will allow us to more clearly analyze feature distortions caused by noise, and will help us to understand the reason for the effectiveness of our post-processing methodology.

2.1 Legend

The following table summarizes the notation used in this paper, providing a quick reference useful when reading later sections.

<table>
<thead>
<tr>
<th>notation</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t$</td>
<td>continuous time</td>
</tr>
<tr>
<td>$s(t)$</td>
<td>clean speech signal</td>
</tr>
<tr>
<td>$n(t)$</td>
<td>additive noise signal</td>
</tr>
<tr>
<td>$h(t)$</td>
<td>convolutional noise signal</td>
</tr>
<tr>
<td>$x(t)$</td>
<td>noise-corrupted speech signal</td>
</tr>
<tr>
<td>$j$</td>
<td>index of a mel-frequency filter</td>
</tr>
<tr>
<td>$J$</td>
<td>number of mel-frequency filters</td>
</tr>
<tr>
<td>$i$</td>
<td>index of a static cepstral coefficient</td>
</tr>
<tr>
<td>$I$</td>
<td>number of static cepstral coefficients</td>
</tr>
<tr>
<td>$d$</td>
<td>index of a feature vector component</td>
</tr>
<tr>
<td>$D$</td>
<td>number of feature vector components</td>
</tr>
<tr>
<td>$P$</td>
<td>power spectrum</td>
</tr>
<tr>
<td>$F$</td>
<td>mel binning filter</td>
</tr>
<tr>
<td>$Q$</td>
<td>mel-frequency spectrum</td>
</tr>
<tr>
<td>$G$</td>
<td>discrete cosine transform matrix</td>
</tr>
<tr>
<td>$C$</td>
<td>(static) cepstral vector</td>
</tr>
<tr>
<td>$\xi$</td>
<td>log energy</td>
</tr>
<tr>
<td>$N$</td>
<td>number of samples in a frame</td>
</tr>
<tr>
<td>$n$</td>
<td>index of discrete time in a frame</td>
</tr>
<tr>
<td>$k$</td>
<td>index of discrete frequency</td>
</tr>
<tr>
<td>$\tau$</td>
<td>frame index</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>noise level</td>
</tr>
</tbody>
</table>

2.2 Different Types of Noises

As is well known, commonly encountered noise can be categorized into additive and convolutional noise. Additive noise is characterized by

$$x(t) = s(t) + n(t),$$

(2)

where $\{s(t)\}$ is the clean speech, $\{n(t)\}$ is the additive noise, and $\{x(t)\}$ is the noisy speech. Convolutional noise is characterized by

$$x(t) = s(t) * h(t),$$

(3)

where $*$ is the convolution operator and $\{h(t)\}$ represents the convolutional noise introduced by the environment (such as reverberation or a microphone). Here the environment is assumed to be time-invariant. In practice, both types of noises are present so the noisy speech signal is

$$x(t) = s(t) * h(t) + n(t).$$

(4)

Note that (4) can be regarded as a special subset of the general distortion, which maps the clean speech into noisy speech,

$$x(t) = \mathcal{F}(s(t)),$$

(5)
where \( F \) defines a general potentially non-linear non-time-invariant mapping. However, this linear time-invariant subset, when applied to a relatively short window of speech, accurately represents a variety of typically encountered noises.

Another source of distortion is not on the produced samples but rather on the production of samples. Under noisy environments, the Lombard effect \([29]\) on articulation, such as elongating the durations of vowels and tilting spectrum to the high frequencies, alters the clean speech signal itself. That is, the realization of speech from a speaker under noisy conditions, \( \{ \tilde{s}(t) \} \), is a distorted version of the realization of speech from the same speaker in clean environments, \( \{ s(t) \} \). This distortion can be viewed as a special case of (5) as well.

### 2.3 Raw Speech Features

For completeness, the definition of standard raw speech features (used as a baseline in our experiments) is given next. Including the basic feature derivation provides a background review, familiarizes the reader with our notation and supplies handy references for later analyses. Those who are familiar with speech features may skip this section and return only later when the equations defined here are referenced.

A Mel-frequency cepstral coefficient (MFCC) vector \([13]\) at a given time frame, \( C \triangleq (C[1] \ldots C[D])^T \), where \( D \) is the number of cepstral coefficients, is defined by

\[
C = G \log_e Q,
\]

(6)

where \( Q \triangleq (Q[1] \ldots Q[J])^T \) is the mel-frequency power spectrum, which is obtained by binning the power spectrum with mel-frequency filter banks, and \( G \) is the \( I \times J \) matrix representing the discrete cosine transform, with

\[
G_{ij} = \sqrt{\frac{2}{J}} \cos \left( \frac{\pi i}{J} (j - 0.5) \right), \quad i = 1 \ldots I, \quad j = 1 \ldots J.
\]

(7)

The MFCC vector is often combined with either the zeroth cepstral coefficient \( C[0] \), defined by

\[
C[0] = \sum_{j=1}^{J} \sqrt{\frac{2}{J}} \log_e Q[j],
\]

(8)

or the log energy \( \xi \), defined by

\[
\xi = \log_e \left( \sum_{n=0}^{N-1} x[n]^2 \right),
\]

(9)

where \( N \) is the number of speech samples in an analysis window (including any zero-padding samples).

### 2.4 Novel MVA Post-Processing Technique

In this section, we define our post-processing technique, consisting of mean subtraction, variance normalization, and ARMA filtering in the cepstral domain. The technique is motivated by considering the distortion of the MFCC features in the presence of additive and convolutional noises, an example of which will be shown shortly and an analysis of which will be given in Section 3. Note in advance that mean subtraction and variance normalization are quite commonly adopted in ASR systems \([23, 30]\). Here we propose ARMA filtering in the cepstral domain, and achieve better performance than mean subtraction and variance normalization alone. Furthermore, we have also found that ARMA filtering outperforms a number of filters derived by other criteria. These results will be presented in Sections 5.1 and 5.3.

Let \( C^{(r)} \) be the feature vector of frame \( r \). Mean subtraction (MS) is defined by

\[
\tilde{C}^{(r)} = C^{(r)} - \mu,
\]

(10)

where \( \mu \) is a mean vector estimated from data and \( \tilde{C} \) is the mean-subtracted feature. Variance normalization (VN) is defined by

\[
\hat{C}^{(r)}[d] = (\sigma^2[d])^{-1/2} \tilde{C}^{(r)}[d],
\]

(11)

\footnote{The frame index \((r)\) will be often omitted for notational clarity, but will be included when needed.}
Figure 1: Block diagram of MVA post-processing technique. $x(t)$: speech waveform; FE: feature extraction; M: mean subtraction; V: variance normalization; A: ARMA filter; BE: back end; $W^*$: recognized text.

where $\tilde{C}$ is the mean-subtracted and variance-normalized feature and $\sigma^2[d]$ is an estimate of the variance of the $d$th component of the feature vector. Our ARMA (Auto-Regression and Moving-Average) filtering is defined by

$$\tilde{C}(\tau) = \frac{\hat{C}(\tau-m) + \cdots + \hat{C}(\tau-1) + \hat{C}(\tau) + \cdots + \hat{C}(\tau+m)}{2m+1},$$

(12)

where $\tilde{C}$ is the MVA post-processed feature and $m$ is the order of the ARMA filter. The special case of $m = 0$ degenerates to no filtering.

The estimation of the mean and the variance, $\mu$ and $\sigma^2$, can be implemented in a number of different ways. In per-side estimation, these quantities are estimated by the entire data in a conversation side [9]. Such estimation is quite robust if the communicating channel is stationary. In on-line estimation [3], they are estimated from only the samples currently available and do not depend on future observations. Such a scheme has low latency and is indispensable for real-time application. Between these two extremes, per-utterance estimation [1], which is adopted in this paper, is defined by

$$\mu = \frac{1}{T} \sum_{\tau=1}^{T} C(\tau),$$

(13)

and

$$\sigma^2[d] = \frac{1}{T} \sum_{\tau=1}^{T} (C(\tau)[d] - \mu[d])^2,$$

(14)

where $T$ is the number of frames in a given utterance.

A block diagram of the post-processing technique is provided in Fig. 1. This method will be referred to as MVA post-processing. Note that MVA post-processing is the same for every feature sequence in every utterance and is not trained on specific data. In other words the post-processing is not tuned to particular types of noise.

Before providing a mathematical analysis of the effects of MVA on speech features, we provide motivation via visual inspection. In Fig. 2, the time sequences of $C[0], C[1]$ and log energy are plotted for speech signals of the utterance of digit string “5376869” corrupted by different levels of additive subway noise from the Aurora 2.0 database. In all three cases ($C[0], C[1],$ and log energy), we see enormous differences between the plots of the clean case and the more noisy cases. In particular, the clean and noisy plots have quite a different average value and dynamic range. After MS and VN is applied, the differences between the clean and noisy cases are made much less severe. In particular, MS and VN combine to bring the time sequences in different noise levels to the same relative average level (via MS) and overall scale (via VN). Still, however, some differences remain between the clean and noisy cases. We notice in particular the case of $C[1]$, that after the application of MS and VN, the time sequences in noisy speech show spurious spikes relative to the time sequences of clean speech. In order to further reduce the differences, we further apply ARMA filtering which smoothes out the sequences thus making them more similar to each other. From a purely visual perspective, the effects of noise on the MVA features are much less severe than either on the raw features or on the MS+VN features. 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 rates (Section 5).

### 3 The Analysis of the Effects of Noises and MVA

Before presenting word-error-rate improvements with our proposed MVA technique, we first spend time analyzing the effects of noise on speech features, both before and after MVA post-processing. This analysis will augment in a
Figure 2: The time sequences of the cepstral coefficient $C[1]$ (top box), $C[0]$ (middle box), and log energy (bottom box), for the digit string “5376869” in clean conditions (first row), 20 dB SNR (second row), 10 dB SNR (third row), 0 dB SNR (fourth row), and -5 dB SNR (last row), without post-processing (left column), with MS and VN (middle column) and with MVA post-processing (right column). The order of the ARMA filter used in this example is $m = 2$. 
Figure 3: The frequency-domain plots of the time sequences of $C[1]$ (top box), $C[0]$ (middle box), and log energy (bottom box), for the digit string “5376869” in clean condition (first row), 20 dB SNR (second row), 10 dB SNR (third row), 0 dB SNR (fourth row), and -5 dB SNR (last row), without post-processing (left column), with MS and VN (middle column), and with MVA post-processing (right column). The x-axis is the frequency axis, ranging from 0 to 50 Hz (Nyquist frequency). The y-axis is the magnitude of spectrum in log scale. The order of the ARMA filter used here is $m = 2$. Note that these are the same examples as in Fig. 2.
mathematical way the simple visual argument given in Fig. 2 showing why we want each of MS, VN, and ARMA, thus providing some theoretical justification to why MVA works as well as it does.

### 3.1 Convolutional Noises and Mean Subtraction

We first analyze the distortion of speech features caused by the convolutional noise and show that mean subtraction is effective in countering such distortion. This analysis is similar to the argument given by Atal [1], that (linear-prediction) cepstral mean subtraction leads to parameters that are invariant in the presence of time-invariant convolutional noise. A succinct account of cepstral distortion in the presence of additive noise and convolutional noise has been given in [42]. Here the analysis is much more detailed and special cautions are taken for the operations of binning and taking logarithms. Such analysis keeps track of the distortion in each domain and helps to understand whether a given processing technique will work or not. In the presence of convolutional noise, recalling that convolution in the time domain leads to multiplication in the frequency domain, the power spectra of \( \{x(t)\} \), \( \{s(t)\} \) and \( \{h(t)\} \) are related by

\[
P_x[k] = P_s[k]P_h[k],
\]

where \( P_x[k] = |X[k]|^2 \), with \( X[k] \) being the discrete Fourier transform of \( x[n] \). The subscript is used to indicate the source. Following (6), the \( i \)th cepstral coefficient of \( x \) is

\[
C_x[i] = \sum_{j=1}^{J} G_{ij} \log_e \left( \sum_{k=0}^{N-1} F_{jk}P_x[k] \right),
\]

where \( F_{jk} \) is the gain of the \( j \)th mel-frequency filter at the \( k \)th frequency. In general, \( C_x \) and \( C_s \) are not simply related in terms of \( h \), because the summation in the argument of the logarithm cannot be factorized. However, if it is assumed that \( P_h \) is relatively flat, such that the variation of the convolutional noise within the pass-band of each mel-frequency filter is small, then

\[
\sum_{k=0}^{N-1} F_{jk}P_x[k] = \sum_{k=0}^{N-1} F_{jk}P_s[k]P_h[k]
\]

\[
\approx P_h[k_j] \sum_{k=0}^{N-1} F_{jk}P_s[k],
\]

where \( P_h[k_j] \) is a representative power spectrum of \( \{h(t)\} \) in the band of filter \( j \). Note that the above assumption does not rule out the possibility of large variation of \( P_h \) among different bands of mel-frequency filters, but only requires that within each band the variation is small. Such assumption tends to be true in the pass-band of a well-designed transmission equipment. However, note that in the case of reverberation, multiple paths/reflections from source to receiver render the frequency response to have peaks and valleys [31], and therefore does not conform to the above assumption.

It follows that

\[
C_x[i] = \sum_{j=1}^{J} G_{ij} \log_e \left( P_h[k_j] \sum_{k=0}^{N-1} F_{jk}P_s[k] \right)
\]

\[
= \sum_{j=1}^{J} G_{ij} \left( \log_e P_h[k_j] + \log_e \left[ \sum_{k=0}^{N-1} F_{jk}P_s[k] \right] \right)
\]

\[
= \sum_{j=1}^{J} G_{ij} \left( \log_e P_h[k_j] + \log_e Q_s[j] \right)
\]

\[
= B_h[i] + C_s[i],
\]

where \( B_h[i] \triangleq \sum_{j=1}^{J} G_{ij} \log_e P_h[k_j] \). Quite similarly, for \( C[0] \),

\[
C_x[0] = \sum_{j=1}^{J} \sqrt{\frac{J}{J}} \left( \log_e P_h[k_j] + \log_e Q_s[j] \right)
\]

\[
= B_h[0] + C_s[0].
\]
physically stem from the configuration of the articulatory organs) renders different patterns in the cepstral domain. Differences in the positions of the formants when speech is mixed with noises from random sources, the overall spectrum tends to be attenuated, leading to the shrinkage of cepstral coefficients (except for the $C(0)$). Therefore the difference between the MFCCs of the noisy and clean speech, $B_h[i]$, depends on the noise $\{h(t)\}$ and, more importantly, does not depend on the speech $\{s(t)\}$. That is, the convolutional noise adds to the feature a bias, the numeric value of which depends on the instantaneous channel characteristics. If it is further assumed that the noise is stationary, e.g., when the environmental configuration is fixed, then it follows from (10), (13), (18) and (19) that for MFCCs,

$$
\begin{align*}
\tilde{C}_x^{(r)}[i] &= C_x^{(r)}[i] - \mu_x[i] \\
&= C_x^{(r)}[i] + B_h[i] - (\mu_s[i] + B_h[i]) \\
&= C_x^{(r)}[i] - \mu_s[i] \\
&= \tilde{C}_s^{(r)}[i], \quad i = 0, 1, \ldots, I.
\end{align*}
$$

One can see that the mean-subtracted features do not change with the presence of a convolutional noise which is stationary and smooth. In our per-utterance framework, if the corrupting noise is of convolutional type and is stationary and spectrally smooth within the duration of the utterance, then MS will work well.

On the other hand, the log energy is

$$
\xi_x = \log_e \left( \sum_{n=0}^{N-1} x[n]^2 \right) \approx \log_e \left( \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2 \right) = \log_e \left( \frac{1}{N} \sum_{k=0}^{N-1} P_x[k]P_h[k] \right),
$$

where the Parseval’s theorem is applied and “$\approx$” indicates the case where pre-emphasis and non-rectangular windows are used on $x[n]$ before calculating $X[k]$. Here the summation in the argument of the logarithm is over the entire spectral range so one can no longer simplify it by pulling $P_h$ out. The bias in the log energy caused by a convolutional noise,

$$
\xi_x - \xi_s = \log_e \left( \frac{\sum_k P_h[k]}{\sum_k P_x[k]} \right),
$$

depends on both the noise $\{h(t)\}$ and speech $\{s(t)\}$. MS doesn’t quite work for this feature because the bias, being dependent on the instantaneous speech, varies from phone to phone.

To summarize, from (20) and (22), we expect MS to benefit more when $C[0]$ is used than when the log energy is used. This point is experimentally verified, as shown in Section 5.1.

### 3.2 Additive Noises and Variance Normalization

Next the distortion of MFCC and log energy features caused by additive noise will be analyzed. This helps to identify the necessary components of an effective post-processing scheme. The LPC cepstral norm shrinkage in the presence of additive white noises is studied in [28]. Insight can be gained by considering the meaning of mel-frequency cepstral coefficients. They are the Fourier coefficients of a sequence, the log mel-spectrum, indexed by the mel frequencies. That is, $C[0]$ is the average of this sequence, and $C[n]$ represents the amplitude of the $n$th harmonic in this sequence. Therefore the $C[n]$’s, in a loose sense, jointly represent the peaks and valleys in the spectrum. The existence of differences in the positions of the formants between different phonemes (representing peaks in the spectrum and physically stem from the configuration of the articulatory organs) renders different patterns in the cepstral domain. When speech is mixed with noises from random sources, the overall spectrum tends to be flattened, leading to the shrinkage of cepstral coefficients (except for $C[0]$).

In order to analyze the distortion of speech features in additive noise, we re-write (2) by

$$
x(t; \gamma) = s(t) + n(t; \gamma) = s(t) + \gamma n_0(t),
$$

where the additive noise $n(t; \gamma) \equiv \gamma n_0(t)$ is parameterized by $\gamma$, the noise level. We analyze just the noise first and...
then return to the signal below. The log mel-spectra of $n(t; \gamma)$ and $n_0(t)$ are related by

$$
\log_e Q_{n(\gamma)}[j] = \log_e \left( \sum_{k=0}^{N-1} F_{jk} (\gamma^2 P_{n_0}[k]) \right)
= 2 \log_e |\gamma| + \log_e \left( \sum_{k=0}^{N-1} F_{jk} P_{n_0}[k] \right)
= 2 \log_e |\gamma| + \log_e Q_{n_0}[j],
$$

(24)

where $Q_{n(\gamma)}$ and $Q_{n_0}$ are the mel-frequency spectra of $n(t; \gamma)$ and $n_0(t)$, respectively. It follows that the cepstral coefficients are related by

$$
C_{n(\gamma)}[i] = \sum_{j=1}^{J} G_{ij} (2 \log_e |\gamma| + \log_e Q_{n_0}[j])
= C_{n_0}[i], \quad i = 1 \ldots I,
$$

(25)

where $C_{n(\gamma)}$ and $C_{n_0}$ are the cepstra of $n(t; \gamma)$ and $n_0(t)$, respectively. It can be seen that MFCC (excluding $C[0]$) is invariant with respect to signal level. For $C[0]$,

$$
C_{n(\gamma)}[0] = \sum_{j=1}^{J} (2 \log_e |\gamma| + \log_e Q_{n_0}[j])
= 2 \sqrt{2} J \log_e |\gamma| + C_{n_0}[0],
$$

(26)

so a change in the signal level only shifts the value of $C[0]$ by a constant. For the log energy,

$$
\xi_{n(\gamma)} = 2 \log_e |\gamma| + \xi_{n_0},
$$

(27)

which has a form similar to $C[0]$. Incidentally, one can see that MS counters the feature distortion caused by signal level variation, from equations (25), (26) and (27).

Following (23), the power spectrum of the noisy speech is

$$
P_{x(\gamma)}[k] = |S[k]|^2 + 2|\gamma|S[k]N_0[k] + \gamma^2 |N_0[k]|^2
= P_s[k] + 2|\gamma|S[k]N_0[k] + \gamma^2 P_{n_0}[k],
$$

(28)

where $P_{x(\gamma)}$, $P_s$ and $P_{n_0}$ are respectively the power spectra of $x(t; \gamma)$, $s(t)$ and $n_0(t)$. It follows, as mel-binning is a linear operation, that

$$
Q_{x(\gamma)}[j] = Q_s[j] + 2|\gamma|Q_1[j] + \gamma^2 Q_{n_0}[j],
$$

(29)

where $Q_1[j] \triangleq \sum_{k=0}^{N-1} F_{jk} |S[k]N_0[k]|$, and $Q_{x(\gamma)}$, $Q_s$ and $Q_{n_0}$ are respectively the power spectra of $x(t; \gamma)$, $s(t)$ and $n_0(t)$. The distortion of the mel-spectrum consists of the last two terms on the right-hand side: one (first-order) term that is proportional to $\gamma$ and depends on both the noise and the speech signal, and the other (second-order) term that is proportional to $\gamma^2$ and depends only on the noise. We first give the distortion of speech features where no assumption about noise level $\gamma$ is made. We then proceed to the cases of low noises ($|\gamma| \ll 1$) and high noises ($|\gamma| \gg 1$) where distortion terms can be simplified through approximation.

### 3.2.1 General Noise Level

In this section, we make no assumptions about $\gamma$ and keep all terms in analyzing the distortion of speech features. In other words, the analysis here applies to all $\gamma$. We will look at special regions of $\gamma$ where we can make simplifying approximations in Sections 3.2.2 and 3.2.3.
By (6) and (29),

\[ C_{x(\gamma)}[i] = \sum_{j=1}^{J} G_{ij} \log e Q_{x(\gamma)}[j] \]

\[ = \sum_{j=1}^{J} G_{ij} \log e \left( Q_s[j] + 2\gamma Q_1[j] + \gamma^2 Q_n_0[j] \right) \]

\[ = \sum_{j=1}^{J} G_{ij} \log e \left( Q_s[j](1 + 2\gamma \frac{Q_1[j]}{Q_s[j]} + \gamma^2 \frac{Q_n_0[j]}{Q_s[j]}) \right) \]

\[ = C_s[i] + \sum_{j=1}^{J} G_{ij} \log e \left( 1 + 2\gamma \frac{Q_1[j]}{Q_s[j]} + \gamma^2 \frac{Q_n_0[j]}{Q_s[j]} \right) \]

(30)

and the distortion is

\[ \delta C_{x(\gamma)}[i] \triangleq \sum_{j=1}^{J} G_{ij} \log e \left( 1 + 2\gamma \frac{Q_1[j]}{Q_s[j]} + \gamma^2 \frac{Q_n_0[j]}{Q_s[j]} \right) \] (31)

where is the distortion, which depends on both the speech \( s(t) \) and the noise \( n(t; \gamma) \). Similarly for \( C[0] \),

\[ C_{x(\gamma)}[0] = \sum_{j=1}^{J} \sqrt{\frac{2}{J}} \log e Q_{x(\gamma)}[j] \]

\[ = C_s[0] + \delta C_{x(\gamma)}[0], \]

(32)

and the distortion is

\[ \delta C_{x(\gamma)}[0] \triangleq \sum_{j=1}^{J} \sqrt{\frac{2}{J}} \log e \left( 1 + 2\gamma \frac{Q_1[j]}{Q_s[j]} + \gamma^2 \frac{Q_n_0[j]}{Q_s[j]} \right) \]. (33)

For the log energy,

\[ \xi_{x(\gamma)} = \log e \left[ \sum_{n=0}^{N-1} \left( s(n) + \gamma n_0(n) \right)^2 \right] \]

\[ = \log e \left[ \sum_{n=0}^{N-1} \left( s(n)^2 + 2\gamma n_0(n)s(n) + \gamma^2 n_0(n)^2 \right) \right] \]

\[ = \log e \left( \sum_{n=0}^{N-1} s(n)^2 \right) \]

\[ + \log e \left( 1 + 2\gamma \frac{\sum_n n_0(n)s(n)}{\sum_n s(n)^2} + \gamma^2 \frac{\sum_n n_0(n)^2}{\sum_n s(n)^2} \right) \]

\[ = \xi_s + \delta \xi_{x(\gamma)}, \]

with distortion

\[ \delta \xi_{x(\gamma)} \triangleq \log e \left( 1 + 2\gamma \frac{\sum_n n_0(n)s(n)}{\sum_n s(n)^2} + \gamma^2 \frac{\sum_n n_0(n)^2}{\sum_n s(n)^2} \right). \] (35)

It is seen from (31), (33) and (35) that the distortion caused by a general additive noise depends on the speech, noise type, and noise level in a complicated way. It is therefore rather difficult to design a general scheme to process corrupted features to clean features. When side-by-side noisy/clean data samples are available, it may be possible to design a potentially non-linear transform to map noisy features to clean features to counter such distortions.
3.2.2 Low Additive Noises

With $|\gamma| \ll 1$, the second-order term in (31) can be neglected and the distortion is given by

$$\delta C_x(\gamma)[i] \approx \sum_{j=1}^{J} G_{ij} \log_e \left( 1 + 2\gamma \frac{Q_1[j]}{Q_s[j]} \right),$$

$$\approx 2\gamma \sum_{j=1}^{J} G_{ij} Q_1[j]$$

$$= 2\gamma C_e[i], \ i = 1, \ldots, I,$$

where $C_e[i] \triangleq \sum_{j=1}^{J} G_{ij} Q_1[j]$, and $\log_e(1 + x) \approx x$ when $|x| \ll 1$. Similarly for $C[0]$, from (33),

$$\delta C_x(\gamma)[0] \approx 2\gamma C_e[0],$$

$$\delta \xi_{x(\gamma)} \approx \log_e \left( 1 + 2\gamma \frac{\sum_n n_0(n)s(n)}{\sum_n s(n)^2} \right),$$

$$\approx 2\gamma \frac{\sum_n n_0(n)s(n)}{\sum_n s(n)^2}$$

$$= 2\gamma e^{-\xi_s} \sum_n n_0(n)s(n),$$

since $\frac{1}{\sum_n s(n)^2} = e^{-\log_e(\sum_n s(n)^2)} = e^{-\xi_s}, \gamma^2 \approx 0$ and $\log_e(1 + x) \approx x$. As the bias is proportional to $e^{-\xi_s}$, it is small when the speech energy is high. This is a desired property and is the reason that log energy feature works better than $C[0]$ if no enhancement or post-processing schemes are adopted. We have experimentally verified this point by comparing the results of using $C[0]$ vs. log energy with the raw feature (without any post-processing) when the noise is relatively low (see Section 5.1).

3.2.3 High Additive Noises

With $|\gamma| \gg 1$, (29) can be approximated by

$$Q_x(\gamma)[j] \approx \gamma^2 (Q_{n0}[j] + \frac{2}{\gamma} Q_1[j]),$$

and the distorted MFCC features become

$$C_x(\gamma)[i] \approx \sum_{j=1}^{J} G_{ij} \log_e \left( \gamma^2 (Q_{n0}[j] + \frac{2}{\gamma} Q_1[j]) \right)$$

$$= \sum_{j=1}^{J} G_{ij} \log_e \left( \gamma^2 Q_{n0}[j](1 + \frac{2}{\gamma} \frac{Q_1[j]}{Q_{n0}[j]}) \right)$$

$$\approx \sum_{j=1}^{J} G_{ij} \left( 2\log_e |\gamma| + \log_e Q_{n0}[j] + \frac{2}{\gamma} \frac{Q_1[j]}{Q_{n0}[j]} \right)$$

$$= C_{n0}[i] + \frac{2}{\gamma} C_e[0], \ i = 1, \ldots, I,$$
where \( C_{e2}[i] = \sum_{j=1}^{J} G_{ij} \frac{Q_{1}[j]}{Q_{ea}[j]} \). For \( C[0] \),
\[
C_x(\gamma)[0] \approx \sum_{j=1}^{J} \sqrt{\frac{2}{J}} \log_e \left( \gamma^2 \left( \frac{Q_{ea}[j]}{Q_{n0}[j]} \right) + C_{n0}[0] + \frac{2}{\gamma} C_{e2}[0] \right)
\]
(41)
where \( C_{e2}[0] = \sum_{j=1}^{J} \sqrt{\frac{2}{J}} \frac{Q_{1}[j]}{Q_{ea}[j]} \). The cepstrum will be mainly decided by the noise \( n_0(t) \), and the contribution from speech signal \( s(t) \) is inversely proportional to \( \gamma \) through the term of \( C_{e2} \). The distortion in the cepstral features is not merely a bias term. That is, even after applying the MS to eliminate \( C_{rna}[i] \) in equations (40) and (41), the remaining term is not the same as that from the clean speech, due to the scale-down of \( \gamma^{-1} \) and that \( C_{e2} \) is different from \( C_{s} \).

Applying VN can mend the first mismatch (scale-down) but can not mend the second (\( C_{e2} \neq C_{s} \)). As mentioned earlier, one way to deal with the difference between clean and noisy speech features is to include the noisy speech samples in the training data set, while another is to use a non-linear transform if side-by-side noisy and clean speech sample pairs are available. In addition, for both methods to work, the noisy data in the training set has to match the test data. Note that if VN is applied alone, \( C_{rna}[i] \) is not eliminated, so the processed features are not close to those from clean speech, and the performance is not expected to improve. This is verified in Section 5.2.

For the log energy
\[
\xi_x(\gamma) = \log_e \left[ \sum_{n=0}^{N-1} (s(n) + \gamma n_0(n))^2 \right]
\]
\[
\approx \log_e \left[ \sum_{n=0}^{N-1} \left( 2\gamma n_0(n)s(n) + \gamma^2 n_0(n)^2 \right) \right]
\]
\[
= \log_e \left[ \gamma^2 \left( \sum_{n=0}^{N-1} n_0(n)^2 \right) \left( 1 + \frac{2}{\gamma} \frac{\sum n_0(n)s(n)}{\sum n_0(n)^2} \right) \right]
\]
(42)
\[
\approx 2 \log_e |\gamma| + \xi_{n0} + \frac{2}{\gamma} \frac{\sum n_0(n)s(n)}{\sum n_0(n)^2}
\]
\[
= 2 \log_e |\gamma| + \xi_{n0} + \frac{2}{\gamma} e^{-\xi_{n0}} \sum n_0(n)s(n),
\]
where \( \xi_{n0} \) is the log energy of \( n_0(t) \). The distortion in (42) is quite similar to (41), so similar arguments apply.

To summarize, additive noise does not contribute an additive bias that can be fully characterized by the noise. Instead, the bias is jointly decided by both speech and noise in ways that are difficult to reverse. In low-noise cases, this term is small and it is shown that log energy is more robust than MFCC. In high-noise cases, after removing terms coming from noise signal, one can apply variance normalization to counter the effect that the feature magnitude is diminished by a factor that is inversely proportional to the noise level.

### 3.3 ARMA Filtering

The human speech is anatomically constrained. To produce a sequence of target phones, the vocal tract system cannot change its configuration arbitrarily fast. It has been argued and demonstrated that the important information in human speech is contained in the low-frequency part of spectral modulation [22, 27]. Since the MFCCs jointly represent the vocal tract configuration (cf. Section 3.2), we believe that the low-frequency parts of the time sequence are more important than the high-frequency parts.

To illustrate, an example is given in Fig. 3. Here the log-scale magnitudes of the discrete Fourier transforms of MFCC and log energy sequences are plotted against the physical frequency. Note that the Nyquist frequency is 50 Hz as there are 100 frames per second.

One can see for the raw features, an increase in the noise level leads to a downward shift in the spectral magnitudes and a flatness in the spectral shape. The MV features are able to counter the effect of downward shift, but it did not help the flatness issue. The MVA features, on the other hand, due to the emphasis in the low-frequency part and de-emphasis in the high-frequency part of the ARMA filter, are able to make the overall frequency plots of clean and noisy
samples quite similar. To be more quantitative, we calculate the Euclidean distances between clean and noisy spectral plots. These numbers are summarized in Table 1. One can see that for each feature, the Euclidean distance between a given noisy sample and the corresponding clean sample increases as the noise level increases (from 20 dB to -5 dB). One can also see that MV features significantly reduce the Euclidean distance relative to raw features between a noisy sample and the corresponding clean sample. Finally, one can see that the application of ARMA filter further reduces the Euclidean distances. Again, visual similarity of features is no guarantee that word error will be reduced, and more importantly increasing similarity at the feature level might actually increase WER because of decreased discrimination ability. Nevertheless, our ARMA filtering in the cepstral domain appears to possess the right balance between feature and speaker normalization without any “word normalization.” This is backed up by the significant error reductions we see as shown in Section 5.

4  Experiments

4.1  Speech Databases

In this section, we evaluate the methodology presented above on two modern noisy-speech corpora, namely the Aurora 2.0 and 3.0 databases [26, 35].

On the Aurora 2.0 database, two training sets and three test sets are defined. The multi-train set consists of both clean and noisy speech, while the clean-train set consists only of clean speech. Test set A is composed of speech with the same types of additive noises as those of the multi-train set, test set B is composed of speech with non-matched additive noises, and test set C is composed of speech with partially matched additive noises and non-matched convolutional noise. Such a partition of the data creates six different degrees of match between training and test data. We will loosely identify multi-train as matched train/test conditions and clean-train as mismatched train/test conditions. On Aurora 2.0, the corruption of speech by noise is done artificially by adding or convolving the clean speech signals with pre-recorded noise signals or channel characteristics.

The Aurora 3.0 database consists of four languages, Danish, Finnish, German and Spanish. The utterances of connected digits are recorded in moving cars using close-talking (CT) and hands-free (HF) microphones. Recording conditions include quiet, low noisy, and high noisy conditions. For each language, three ASR tasks are defined. The well-matched (WM) task is to train on 70% of both CT and HF recordings from all conditions and test on the remaining 30%. The medium-mismatched (MM) task is to train on 70% of CT recordings from all conditions and test on the remaining 30% of the HF recordings from low and high noisy conditions. The highly-mismatched (HM) task is to train on 70% of CT recordings from all conditions and test on 30% of the HF recordings from low and high noisy conditions. Note that the utterances

Table 1: The Euclidean distances in the frequency domain between the clean sample and the noisy samples. For example, the number 919 in position (C[1] raw, 20 dB) is the Euclidean distance between the C[1] features (without any post-processing) of clean and 20 dB samples. Here the distance is defined as \( \left( \sum_{n=1}^{T} (X_{c1}[n] - X_{c2}[n])^2 \right)^{1/2} \), where \( n \) is the discrete frequency index and subscript \( c_1 \) and \( c_2 \) represent clean and noisy examples respectively. Note that here the original magnitude is used instead of the log scale in calculating the distance, unlike in Fig. 3.

<table>
<thead>
<tr>
<th></th>
<th>20 dB</th>
<th>10 dB</th>
<th>0 dB</th>
<th>-5 dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>C[1] raw</td>
<td>919</td>
<td>1339</td>
<td>1890</td>
<td>1922</td>
</tr>
<tr>
<td>C[1] MV</td>
<td>83</td>
<td>128</td>
<td>176</td>
<td>213</td>
</tr>
<tr>
<td>C[1] MVA</td>
<td>71</td>
<td>110</td>
<td>147</td>
<td>171</td>
</tr>
<tr>
<td>C[0] raw</td>
<td>3116</td>
<td>4669</td>
<td>6219</td>
<td>7014</td>
</tr>
<tr>
<td>C[0] MV</td>
<td>69</td>
<td>142</td>
<td>161</td>
<td>219</td>
</tr>
<tr>
<td>C[0] MVA</td>
<td>61</td>
<td>132</td>
<td>148</td>
<td>210</td>
</tr>
<tr>
<td>( \xi ) raw</td>
<td>842</td>
<td>1278</td>
<td>1685</td>
<td>1918</td>
</tr>
<tr>
<td>( \xi ) MV</td>
<td>74</td>
<td>137</td>
<td>151</td>
<td>202</td>
</tr>
<tr>
<td>( \xi ) MVA</td>
<td>67</td>
<td>127</td>
<td>141</td>
<td>194</td>
</tr>
</tbody>
</table>
Table 2: Experimented front-end configurations. Q1: Use $C[0]$ vs. log Energy. Q2: Apply MVA only on static features vs. on both static and dynamic features. Q3: number of mel-frequency bins are used during feature extraction. Q4: band-limited spectrum vs. the entire spectrum from 0 to the Nyquist frequency.

<table>
<thead>
<tr>
<th>Config.</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>logE</td>
<td>static</td>
<td>23 channels</td>
<td>band-limited</td>
</tr>
<tr>
<td>2</td>
<td>$C[0]$</td>
<td>static</td>
<td>23 channels</td>
<td>band-limited</td>
</tr>
<tr>
<td>3</td>
<td>$C[0]$</td>
<td>static+dynamic</td>
<td>23 channels</td>
<td>band-limited</td>
</tr>
<tr>
<td>4</td>
<td>$C[0]$</td>
<td>static+dynamic</td>
<td>26 channels</td>
<td>entire</td>
</tr>
</tbody>
</table>

in the Aurora 3.0 database are recorded in real noisy environments, and may include Lombard effects [29].

4.2 System Setup

4.2.1 Front-End Configuration

In evaluating speech features, we investigate four different front-end configurations for the Aurora 2.0 database, with three of them further used on Aurora 3.0. Specifically, Configuration 1 uses the log energy along with $C[1]$ to $C[12]$ as the static feature vector, on which the MVA post-processing is applied, and then the delta and delta-delta features are calculated. In other words, the post-processing is only applied on the static features, and the delta and delta-delta features are computed after MVA post-processing is performed. Configuration 2 is different from Configuration 1 in that $C[0]$ is used instead of log energy. The comparison between Configuration 1 and 2 can verify the prediction that $C[0]$ is better than log energy as we argued in Section 3.1. Configuration 3 is different from Configuration 2 in that the MVA post-processing is applied to both the static and the dynamic features. The comparison between Configuration 2 and 3 is done to see if it improves the performance of applying MVA on dynamic features as well. Configuration 4, which we used in [10], is different from Configuration 3 in that it does not apply a band-pass filter (passing 64-4kHz as in configurations 1-3) to the spectrum, that it has more mel-frequency filters (26 rather than 23), and that the raw feature is already mean-subtracted. This tends to make the performance with raw features better than without mean subtraction. The main differences between these configurations are summarized in Table 2. The first three configurations are designed to compare the effectiveness of $C[0]$ and log energy and the effect of applying MVA to the dynamic features as well as the static features. Configuration 4 is included to evaluate the effect of not using a band-limited speech spectrum and an increased number of mel-frequency filters.

The evaluation of different front-end configurations show that Configuration 3 is overall the best. Therefore we use it in the experiments in evaluating different aspects of MVA, including multi-domain post-processing in Section 5.2, designed filters in Section 5.3, data-driven filters in Section 5.4 and different ARMA orders in Section 5.5.

4.2.2 Back-End Configuration

A whole-word HMM back end is used consistently in the experiments. Specifically, the feature space is 39-dimensional, each word model has 16 emitting states, and each state emitting density function is a Gaussian mixture with 3 components. The silence model has 3 emitting states, each having Gaussian mixture density with 6 components. The short-pause model has only one emitting state which is tied to the middle state of the silence model.

5 Results

A few earlier experimental results have been presented in previous work [10, 11]. In this paper we present many new results. These new experiments are designed to verify the points raised during our theoretical analysis and during the course of experimentation. Specifically, we ask and answer the following questions: “Is $C[0]$ better than log energy and when/where?”, “Is applying MVA to both static and dynamic features crucial”, “Does ARMA filter provide additional gain on top of MS and VN?”, “Is ARMA better than other smoothing filters and what is the optimal order of the ARMA filter?”, “Where and in what order should one apply MS, VN and ARMA?”. We answer these questions one by one in the following sections.
Table 3: Word Accuracies (as percentages) of MVA on Aurora 2.0 with various configurations of front end. The back end is fixed, with 39-dimensional feature vector, 16 states per word and 3 Gaussian components per state. Top: multi-train (matched train/test conditions) task; bottom: clean-train (mismatched train/test conditions) task.

<table>
<thead>
<tr>
<th>Aurora 2.0 multi-train (matched train/test conditions)</th>
<th>0-20 dB</th>
<th>-5 dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>clean raw MV MVA</td>
<td>raw MV MVA</td>
<td>raw MV MVA</td>
</tr>
<tr>
<td>1</td>
<td>98.6</td>
<td>97.9</td>
</tr>
<tr>
<td>2</td>
<td>98.5</td>
<td>98.3</td>
</tr>
<tr>
<td>3</td>
<td>98.5</td>
<td>98.6</td>
</tr>
<tr>
<td>4</td>
<td>98.4</td>
<td>98.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Aurora 2.0 clean-train (mismatched train/test conditions)</th>
<th>0-20 dB</th>
<th>-5 dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>clean raw MV MVA</td>
<td>raw MV MVA</td>
<td>raw MV MVA</td>
</tr>
<tr>
<td>1</td>
<td>99.0</td>
<td>99.0</td>
</tr>
<tr>
<td>2</td>
<td>99.1</td>
<td>99.1</td>
</tr>
<tr>
<td>3</td>
<td>99.0</td>
<td>99.1</td>
</tr>
<tr>
<td>4</td>
<td>99.1</td>
<td>99.2</td>
</tr>
</tbody>
</table>

5.1 Comparison of Different Front-End Configurations

The first set of experiments are conducted to compare different front-end configurations specified in Section 4.2.1. The results are summarized in Table 3 for Aurora 2.0 and Table 4 for Aurora 3.0.

With both multi- and clean-train tasks of Aurora 2.0, MVA post-processing technique results in significant improvements in all configurations inspected. Furthermore, the improvement with MVA processing over without MVA processing is dependent on the raw features. For the multi-train task, the relative improvement with 0 – 20 dB test data using Configuration 3 is better than Configuration 2 (45% vs. 38%), which is better than Configuration 1 (25%). Similarly, for the clean-train task the relative improvements are 65%, 50% and 38% for Configuration 3, 2, and 1 respectively. We also notice that using more mel-frequency bins over the entire spectrum does not lead to better performance, as we compare Configurations 3 and 4. Here the number of mel-frequency bins and the pass-band (entire or limited spectrum) jointly decide the spectral locations of the binning filters which essentially distribute uniformly on the mel-scale across the pass-band.

On Aurora 3.0, with all WM, MM, and HM tasks, similar pattern of performance and improvement has been observed with this real-world noisy speech database. Specifically, the overall relative improvement changes from 35% to 44% with WM tasks, from 40% to 44% with MM tasks and from 52% to 69% with HM tasks, when we change from Configuration 1 to Configuration 2, i.e., from log energy to $C[0]$. Noteworthy is the very big gain in performance in the HM tasks, especially Finnish, with $C[0]$ and MVA.

The following points summarize results in Tables 3 and 4.

- With noisy test data, MVA improves the performance significantly with very little extra computational cost;
- With clean test data, MVA doesn’t hurt the performance when using $C[0]$ (it hurt a little only in the multi-train case using log energy);
- The improvement increases as the noise level increases and/or the degree of mismatch increases;
- Based on these experimental results, one can see that $C[0]$ does provide better performance than log energy, as theoretically predicted in Section 3.1;
- Comparing results of Configurations 2 and 3, one can see that for artificially corrupted noisy speech of Aurora 2.0, applying MVA on both dynamic and static features is significantly better than applying MVA on static features alone. However, on the real-world noisy speech Aurora 3.0, the results are somewhat mixed;
- Comparing results of the MV and MVA columns, one can see that ARMA filtering does provide additional performance gain for noisy test data (91.2% vs. 92.0% with multi-train; 78.4% vs. 83.6% with clean-train). These results are pretty much what we have expected from consideration in Section 3.3;
Table 4: Word accuracies (as percentages) for Aurora 3.0 with end-pointed data. WM: Well-Matched task; MM: Medium-Mismatched task; HM: Highly-Mismatched task.

<table>
<thead>
<tr>
<th></th>
<th>WM</th>
<th>MM</th>
<th>HM</th>
</tr>
</thead>
<tbody>
<tr>
<td>German</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>raw</td>
<td>MV</td>
<td>MVA</td>
</tr>
<tr>
<td>1</td>
<td>91.5</td>
<td>94.7</td>
<td>81.5</td>
</tr>
<tr>
<td>2</td>
<td>92.1</td>
<td>95.0</td>
<td>81.5</td>
</tr>
<tr>
<td>3</td>
<td>91.2</td>
<td>95.3</td>
<td>79.9</td>
</tr>
<tr>
<td>Danish</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>raw</td>
<td>MV</td>
<td>MVA</td>
</tr>
<tr>
<td>1</td>
<td>86.1</td>
<td>90.0</td>
<td>67.7</td>
</tr>
<tr>
<td>2</td>
<td>87.5</td>
<td>92.2</td>
<td>68.4</td>
</tr>
<tr>
<td>3</td>
<td>84.5</td>
<td>91.5</td>
<td>63.3</td>
</tr>
<tr>
<td>Finnish</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>raw</td>
<td>MV</td>
<td>MVA</td>
</tr>
<tr>
<td>1</td>
<td>91.7</td>
<td>93.4</td>
<td>79.8</td>
</tr>
<tr>
<td>2</td>
<td>91.1</td>
<td>93.5</td>
<td>78.6</td>
</tr>
<tr>
<td>3</td>
<td>90.0</td>
<td>91.9</td>
<td>72.0</td>
</tr>
<tr>
<td>Spanish</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>raw</td>
<td>MV</td>
<td>MVA</td>
</tr>
<tr>
<td>1</td>
<td>93.1</td>
<td>95.1</td>
<td>83.1</td>
</tr>
<tr>
<td>2</td>
<td>92.7</td>
<td>95.6</td>
<td>86.0</td>
</tr>
<tr>
<td>3</td>
<td>90.9</td>
<td>95.3</td>
<td>83.3</td>
</tr>
</tbody>
</table>

- Another interesting point is to compare the results of 0 − 20 dB test data of raw features with Configuration 1 and Configuration 2 (85.8% vs. 85.6% with multi-train; 60.0% vs. 57.9% with clean-train). As we have pointed out in Section 3.2.2, with low noise without any post-processing, the log energy can outperform $C[0]$, which is indeed the case here.

5.2 Results of Multi-Domain Post-Processing

Filtering techniques have been proposed in other domains as well, e.g., [24]. In order to better understand the merits of such processing in different domains, we also investigate a multi-domain generalization of the MVA technique by applying MVA post-processing in both the log mel-spectral and the cepstral domain. Furthermore, we also decompose MVA into atomic MS, VN and ARMA modules and apply these modules in different orders. Fig. 4 depicts multi-domain post-processing that the MS, VN, and ARMA modules are optionally inserted in the log mel spectral and the cepstral domain. Specifically, in each domain, we have considered nine alternatives:

1. N – no processing.
2. M – MS.
3. V – VN.
4. A – ARMA.
5. MV – MS followed by VN.
6. MA – MS followed by ARMA.
7. AM – ARMA followed by MS.
8. AMV – ARMA followed by MS, followed by VN.
Figure 4: Block diagram of multi-domain post-processing technique. Here $R$ stands for the logarithm of the output of the mel-frequency binning filters, GPPM represents the general post-processing module that can be switched to one of the options specified in Section 5.2, and $R'$ is the post-processed $R$. DCT stands for discrete cosine transform, and $C''$ is the cepstral features after multi-domain post-processing are performed.

9. MVA – MS followed by VN, followed by ARMA.

Note that the indexes specified here (1 through 9) will be used in Fig. 5–10.

The results of Aurora 2.0 and Aurora 3.0 are depicted in the plots in Fig. 5–10. In the figures, we represent a multi-domain post-processing combination as a grid point specified by $(x, y)$, where $x$ indexes the post-processing in the log mel spectral domain and $y$ indexes the post-processing in the cepstral domain. While the amount of information displayed in these plots is quite large, for completeness and accuracy all plots are given rather than just a subset. Summarizing the information in the plots, a number of conclusions can be drawn as follows.

- VN alone degrades the performance relative to the baseline, as one can see a dip in $(1, 3)$, representing no processing in the log mel-spectral domain and VN in the cepstral domain, in the figures. This has been pointed out in Section 3.2.3.

- One can see that for matched tasks, i.e., WM in Aurora 3.0 and multi-train in Aurora 2.0, the variation of performance with respect to different processing techniques are smaller than mismatched tasks. This shows that it is important to choose correctly the post-processing scheme especially in mismatched tasks. Wrong types of post processing will severely degrade performance.

- Comparing $(1, 9)$ and $(9, 1)$, one can see that MVA in the cepstral domain is better than in the log mel spectral domain. Again, the difference is bigger in mis-matched tasks than in matched tasks. Therefore, it is important to apply MVA in the right domain.

- There is no large effect of switching MVA and AMV in the cepstral domain, even though the two are not commutative. In the log mel-spectral domain, the difference between MVA and AMV is larger.

5.3 Comparison with Designed Filters

In order to see whether there is an intrinsic advantage to ARMA filter, or if any filter suffices, we design FIR filters that have the same complexity as the ARMA filter. Specifically, these FIR filters are designed to have 5 taps, linear phase responses, and magnitude responses with least squared error. They are used to replace ARMA filter to post-process the feature streams after MS and VN have been applied. These filters have the same normalized bandwidth, with pass-bands corresponding to low-pass filter, band-pass filters and high-pass filter specified as follows.

1. low–pass filter with pass-band $[0.0, 0.6]$ in normalized frequency.
2. high-pass filter with pass-band $[0.4, 1.0]$.
3. band-pass filter with pass-band $[0.1, 0.7]$.
4. band-pass filter with pass-band $[0.2, 0.8]$.
5. band-pass filter with pass-band $[0.3, 0.9]$.
6. baseline, i.e., without MS, VN or filtering.
7. MV, i.e., without filtering.
Figure 5: Word accuracies (as percentages) for Aurora 2.0 multi-train (matched train/test conditions) with MVA processing in both log mel spectral and cepstral domains. The x-axis (going northwest-southeast) represents processing in the log mel-spectral domain, while the y-axis (going southwest-northeast) represents processing in the cepstral domain. The grid points on the x- and y- axes correspond to the post-processing defined in Section 5.2. Top: clean test data; center: 0 – 20 dB test data; bottom: −5 dB test data.
Figure 6: Word accuracies (as percentages) for Aurora 2.0 clean-train (mismatched train/test conditions) task with MVA processing in both log mel spectral and cepstral domains. Top: clean test data; center: 0 – 20 dB test data; bottom: −5 dB test data.
Figure 7: Word accuracies (as percentages) for Aurora 3.0 German with MVA processing in both log mel spectral and cepstral domains. Top: WM task; center: MM task; bottom: HM task.
Figure 8: Word accuracies (as percentages) for Aurora 3.0 Danish with MVA processing in both log mel spectral and cepstral domains. Top: WM task; center: MM task; bottom: HM task.
Figure 9: Word accuracies (as percentages) for Aurora 3.0 Finnish with MVA processing in both log mel spectral and cepstral domains. Top: WM task; center: MM task; bottom: HM task.
Figure 10: Word accuracies (as percentages) for Aurora 3.0 Spanish with MVA processing in both log mel spectral and cepstral domains. Top: WM task; center: MM task; bottom: HM task.
The results are summarized in Fig. 11-14. One can see that as the pass-band goes from low frequency to high frequency, the performance systematically degrades, confirming the common belief that important information of speech is in the low frequency part of the modulation spectrum [22]. Note that although the best designed filter (the low-pass one) has a performance close to that of the ARMA filter, the former is still not as good as the latter. Another advantage of the ARMA filter is that it can be implemented entirely in place (only a single scalar temporary variable is needed to implement the ARMA filter, whereas an array of temporaries one-half the length of the FIR filter is needed in the FIR case).

5.4 Comparison with Data-Driven Filters

To further investigate the advantage of using ARMA filter, we next investigate if a data-designed filter has the ability to beat the simple ARMA filter. To keep the same complexity, a five-tap data-driven filter minimizing the squared error between filtered noisy cepstrum and clean cepstrum is applied to the feature sequence after MS and VN. Specifically,
we solve the optimization problem

$$\min_a \sum_{u=1}^U \sum_{\tau=1}^{T_u} \left( X_u^{(\tau)} - \sum_i (a_i Y_u^{(\tau+i)}) \right)^2, \quad i = -2, \ldots, 2,$$

(43)

where \(a = \{a_i\}\) is the set of filter coefficients, \(U\) is the number of utterance pairs, \(T_u\) is the number of frames in utterance \(u\), and \(X_u\) and \(Y_u\) are the feature sequences corresponding to an utterance pair in clean and noisy environments. Define the following quantities,

$$r_{XY}(i) \triangleq \frac{\sum_u \sum_{\tau} X_u^{(\tau)} \cdot Y_u^{(\tau+i)}}{\sum_u \sum_{\tau} \sum_{i}}, \quad i = -2, \ldots, 2$$

$$r_{YY}(i) \triangleq \frac{\sum_u \sum_{\tau} Y_u^{(\tau)} \cdot Y_u^{(\tau+i)}}{\sum_u \sum_{\tau}}, \quad i = 0, \ldots, 4,$$

(44)

where \(r_{XY}\) is the cross-correlation of \(X\) and \(Y\) and \(r_{YY}\) is the auto-correlation of \(Y\). Then the solution of (43) is

$$a = R^{-1} * r_{XY},$$

(45)

where \(R\) is the Toeplitz matrix formed from \(r_{YY}\). The optimal 5-tap non-causal FIR filter from noisy cepstrum to clean cepstrum found with 8440 pairs of utterance pairs on Aurora 2.0 is \(a = \{0.1090, 0.1265, 0.4003, 0.1667, 0.0127\}\). Interestingly, the frequency response of this filter bears a strong similarity to ARMA filter, in the sense that both are low pass filters and the main lobe of the frequency response extends from dc to about 0.6. We also experimented the case when each component in the feature vector is derived its own filter, and we still observe similar frequency responses for these component-wise filters. Experimental results with the data-driven filters are summarized in Table 5. Surprisingly, it is still the ARMA filter that outperforms the others.

### 5.5 Comparison of ARMA Orders

Next we investigate if the order of the ARMA filter (i.e. \(m\)) has a large effect on the performance, since there is an inherent trade-off in choosing the order. That is, 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 order will be a small positive integer [10].

The experimental results are summarized in Fig. 15-20. From these graphs, one can see that other than the very noisy test data in Aurora 2.0 and HM tasks in Aurora 3.0, the optimal ARMA order is often a small integer. For the very noisy test data in Aurora 2.0 and HM tasks in Aurora 3.0, the improvement in increasing the order quickly decreases beyond \(m = 2\). This verifies that a small integer, such as \(m = 2\), is a good choice for the order of ARMA. It works well in all cases.

Based on these figures, the ranks of performances of different orders do depend on the tasks. Specifically, when the training/test data are mismatched (such as the HM tasks of Aurora 3.0 and clean train task of Aurora 2.0) or when the test data is very noisy (such as the −5 dB SNR test set), one can see that the introduction of ARMA filter helps and that the effect of changing the order of ARMA filter is quite significant.
Figure 15: Word accuracies (as percentages) for Aurora 2.0 multi-train (matched train/test conditions) task with various orders of the ARMA filter. Group 1 is clean test data, group 2 is 0 – 20 dB test data, and group 3 is -5 dB test data. Within each group, the bars from left to right correspond to order 0, 1, 2, 3, 4 respectively.

Figure 16: Word accuracies (as percentages) for Aurora 2.0 clean-train (mismatched train/test conditions) task with various orders of the ARMA filter. Same legend as in Fig. 15.

Figure 17: Word accuracies (as percentages) for Aurora 3.0 German with various orders of the ARMA filter. Group 1 is the WM task, group 2 is the MM task, and group 3 is the HM task. Within each group, the bars from left to right correspond to order 0, 1, 2, 3, 4 respectively.

Figure 18: Word accuracies (as percentages) for Aurora 3.0 Danish with various orders of the ARMA filter. Same legend as in Fig. 17.
Figure 19: Word accuracies (as percentages) for Aurora 3.0 Finnish with various orders of the ARMA filter. Same legend as in Fig. 17.

Figure 20: Word accuracies (as percentages) for Aurora 3.0 Spanish with various orders of the ARMA filter. Same legend as in Fig. 17.

Table 5: Word accuracies (as percentages) for Aurora 2.0 with data-driven minimum-squared-error filter. DDT1 is the case where a single filter is derived and used for all cepstral time sequences, while DDT2 is the case where each cepstral coefficient is derived its own filter. In other words, in DDT1 all components use the same filter, while in DDT2 different components use different filters.

<table>
<thead>
<tr>
<th></th>
<th>multi-train</th>
<th>clean-train</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ARMA</td>
<td>DDT1</td>
</tr>
<tr>
<td>clean</td>
<td>98.5</td>
<td>98.5</td>
</tr>
<tr>
<td>0-20dB</td>
<td>92.0</td>
<td>91.9</td>
</tr>
<tr>
<td>-5 dB</td>
<td>42.0</td>
<td>40.6</td>
</tr>
</tbody>
</table>
6 Summary

In this paper, the distortion of MFCC and log energy features is analyzed in the presence of additive and convolutional noises from scratch. Based on the analysis, a post-processing technique, MVA, is implemented to counter such distortion. The experiments are conducted on the Aurora 2.0/3.0 databases. It is verified that the effectiveness depends on the raw feature sets, the applied domains, the smoothing filters and the orders of the filter. We find that if one simply uses a second order MVA with $C[0] \ldots C[12]$ in the cepstral domain (Configuration 3), the improvement over the raw features is 45% in the $0 - 20$ dB test data for the multi-train task and 65% for the clean-train task in Aurora 2.0, and 49% for the WM tasks, 50% for the MM tasks and 72% for the HM tasks in Aurora 3.0. Such performance rivals that by much more complicated systems yet comes without increasing the computational cost over the baseline system.

Since MVA post-processing is performed within the feature space, it is straightforward to combine MVA with other noise-robust techniques. The back end used in this paper is a very simple one. MVA will be further investigated in combination with more refined back ends, e.g. [36, 20, 33, 5], that are often necessary in medium- to large-vocabulary ASR tasks.

References


