LOW COMPLEXITY CONNECTED DIGIT RECOGNITION
FOR MOBILE APPLICATIONS

Prabhu Ragavan         Sanil K. Gupta
Bell Laboratories, Lucent Technologies
101 Crawfords Corner Road
Holmdel NJ 07733-3030

ABSTRACT

For low complexity, mobile, hands-free, speaker independent connected digit recognition, a fixed-point digital signal processor based implementation is essential. In this paper, we investigate algorithms for connected-digit recognition using whole-word digit models and a background model. We show that significant improvement can be achieved by using background model adaptation, continuously adaptive separate cepstral mean subtraction for background and speech segments and discriminative training. The system achieves almost 96% digit accuracy on a 15 speaker database of speech recorded in a car. A real-time system using the Lucent’s DSP1627 has also been developed. We also present the results of our experiments in reducing complexity for the fixed-point system. These include a method to approximate state-likelihood computation using a Vector Quantization based mixture selection and use of beam width pruning during Viterbi decoding.

1 INTRODUCTION

There has been significant research to address the problem of speaker independent automatic speech recognition (ASR) using Hidden Markov Models [1]. The recognition performance of an ASR system is closely tied to the computing resources available, the complexity of the task, and the operating environment.

In this paper, we focus on real-time connected-digit recognition in a mobile environment using a device with limited CPU and memory resources such as a fixed-point Digital Signal Processor (DSP). A mobile environment is characterized by a large variability in noise level. Traditional methods of achieving improved performance for connected-digit recognition in such adverse environments usually impose heavy requirements on memory and processing power [2, 3]. Therefore they may be unsuitable for a fixed-point DSP implementation.

We propose algorithms that are of low complexity and yet achieve enhanced performance. We use whole word digit models that reduce memory requirements as compared to context-dependent models. We focus on achieving improved recognition with these models using techniques such as discriminative training and environment adaptation using the background model. A Vector Quantization (VQ) based mixture selection for state-likelihood computation is also discussed. This reduces computational requirements without significantly compromising recognition performance.

In Section 2, we describe our baseline system and characterize its performance for this task using a speech database collected in a car. Sections 3, 4, and 5 describe algorithmic improvements. Section 6 discusses various methods of reducing computational complexity. An analysis of the performance is presented in Section 7.

2 BASELINE SYSTEM

In our system, we use 11 Hidden Markov Model (HMM) based whole word digit models (0-9 and Oh). The speech is analyzed in 20 ms frames with a 10 ms overlap between successive frames. Each frame is processed to extract 12 Linear Prediction based Cepstral Coefficients, 12 Deltas and one Delta Energy to form the final observation vector of size 25. In the baseline system, no cepstral mean subtraction is performed. Each digit model has 16 states with 8 mixture components per state. A single state 16 mixture component background model is used for silence and other non-speech segments.

The baseline models are trained on the TI-DIGITS training database with car noise added to it in the sample domain. The car noise was collected in a car driven at various speeds. This noise was added at an average SNR of 5dB to two-thirds of the training database and at an average SNR of 25dB to the rest of the database.

Figure 1. Distribution of SNR of TI-DIGITS training database (dash) and the estimated SNR of the test database (solid).
The models were evaluated on a 15 speaker database, collected in a car, with approximately 1200 sentences, that had an average estimated SNR of 1.9 dB. The average digit-string length of the test database is 5.6. Figure 1 shows the distribution of the SNR in the training database and the test databases.

Table 1 shows the performance of the baseline system. It is clear that the performance is poor. The rest of the paper discusses various methods to improve performance.

<table>
<thead>
<tr>
<th>Digit Error</th>
<th>Insertion</th>
<th>Deletion</th>
<th>Substitution</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.8%</td>
<td>1.9%</td>
<td>4.9%</td>
<td>9.0%</td>
</tr>
</tbody>
</table>

Table 1. Baseline performance

3 CEPSTRAL MEAN SUBTRACTION

Cepstral Mean Subtraction (CMS) attempts to reduce channel distortion effects by computing the time-invariant part of the channel transfer function which is additive in the cepstral domain.

In our real-time system, the cepstral mean is updated using a leaky integrator with a time constant of approximately 20 observations. Thus the current estimate of the cepstral mean, $\mathbf{c}_t$, is given by,

$$\mathbf{c}_t = \gamma \mathbf{c}_t + (1 - \gamma) \mathbf{c}_{t-1},$$

where $\mathbf{c}_t$ is the cepstrum at time $t$ and $\gamma$ is the CMS integration constant. The compensated cepstrum $\mathbf{c}'_t$ is given by,

$$\mathbf{c}'_t = \mathbf{c}_t - \bar{\mathbf{c}}_t.$$ (2)

It has been shown that the use of separate means for speech and background segments of the speech can significantly improve performance, particularly for noisy environments [4]. In this paper, we use an energy-based voice activity detector that classifies observations as speech or background. Two separate cepstral mean vectors, a speech mean, $\mathbf{c}'_s$, and a background mean, $\mathbf{c}'_b$, are updated using leaky integration.

The compensated cepstrum $\mathbf{c}'_t$ for 2-Level CMS is computed as,

$$\mathbf{c}'_t = \begin{cases} 
\mathbf{c}_t - \bar{\mathbf{c}}_t & \text{for speech} \\
\mathbf{c}_t - \bar{\mathbf{c}}_t & \text{for background} 
\end{cases}.$$ (3)

Table 2 shows the results of performing Single and 2-Level CMS on the input observation. The Single CMS improves the performance from 15.8% to 7.90%, a reduction of 52%. A further reduction of 30% is obtained from this error using 2-Level CMS.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Single CMS</td>
<td>7.6%</td>
<td>0.7%</td>
<td>1.8%</td>
<td>5.0%</td>
</tr>
<tr>
<td>2-Level CMS</td>
<td>4.8%</td>
<td>0.4%</td>
<td>1.8%</td>
<td>2.6%</td>
</tr>
</tbody>
</table>

Table 2. Results of performing Single and 2-Level CMS on input observation

4 BACKGROUND MODEL ADAPTATION

Another technique used to improve performance is the continuous adaptation of the background model. Most adaptation techniques attempt to optimize a model’s parameters to better reflect the adaptation data. For this, an alignment step needs to be performed to find the association of the adaptation data to the model state. Typically this is a post processing step that is computationally intensive and also requiring additional storage. Our background model, being a single state model, requires no computation to assess the observation-state association.

For each input observation that is classified as background by the voice activity detector, the mixture component nearest to the observation is determined based on an euclidean divergence measure. The mean vector for this mixture component is then adapted using leaky integration,

$$\mu_t = \beta \mu_{t-1} + (1 - \beta) \mathbf{a}_t,$$ (4)

where $\mu_t$ is the background model mean, $\mathbf{a}_t$ is the current observation that is classified as background and $\beta$ is the leaky integration time constant.

Table 3 shows the result of using 2-Level CMS with background model adaptation on the test database. A reduction of 3.7% in digit error is observed. We have observed that background adaptation is most effective when there is a mismatch between training and test conditions.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4.7%</td>
<td>0.3%</td>
<td>2.0%</td>
<td>2.5%</td>
</tr>
</tbody>
</table>

Table 3. Effect of Background Model adaptation using the 2-Level CMS model

5 DISCRIMINATIVE TRAINING

The discriminative training algorithm attempts to minimize the classification error given the speech training data [5]. This is in contrast with traditional Maximum Likelihood (ML) training that maximizes the likelihood of the model given the training data. The problem of designing an optimal classifier now becomes the problem of finding the right parameter set that effectively discriminates the observations to achieve best classification instead of optimizing the model distributions to fit the data.

Table 4 shows the effect of using discriminative training on the digit recognition error. The reduction in digit error is 6.25% compared to Table 2.

Table 5 shows the effect of using discriminative training and background model adaptation on the performance. The reduction in digit error compared to the performance of just 2-Level CMS in Table 2 is 10.4%.

6 COMPLEXITY REDUCTION

We have implemented our system on the Lucent DSP1627 fixed-point processor. Table 6 shows the performance of the fixed-point implementation with 2-Level CMS and background model adaptation. This result can be compared to the performance in Table 3. It can be seen that the two results are almost identical with no effect of fixed-point implementation on the performance.
Table 4. Results of using Discriminative training and 2-Level CMS.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5%</td>
<td>0.8%</td>
<td>1.1%</td>
<td>2.6%</td>
</tr>
</tbody>
</table>

Table 5. Results of performing Discriminative training, Background model adaptation and 2-Level CMS.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2%</td>
<td>0.9%</td>
<td>1.2%</td>
<td>2.5%</td>
</tr>
</tbody>
</table>

6.1 Viterbi pruning beam width
An effective method of reducing computations for recognition is to reduce the number of active search paths during Viterbi. This can be controlled by setting a threshold on the current accumulated likelihood of a search path. This threshold, called the beam width, is usually set relative to the maximum accumulated likelihood along any search path.

Table 7 shows the performance of the fixed-point system as a function of beam width. Column 2 shows the average % CPU utilization for an utterance with different values of the Viterbi pruning beam width. Column 3 shows the effect of using discriminative training with 2-Level CMS and background model adaptation for these beam widths on our test database. There is no increase in digit error but a substantial reduction in CPU utilization (a relative reduction of 31%).

6.2 VQ based state-likelihood computation
In small vocabulary ASR systems, the state likelihood computations consume most CPU resources. In order to reduce the CPU usage, we use a VQ based mixture selection to approximate the state-likelihoods that requires computation of fewer mixture probabilities [6]. In this method, a set of VQ clusters is computed by clustering the mixture components in the model set. For clustering, the measure of divergence between a pair of mixture components is a Mahalanobis-like weighted Euclidean distance between the mean vectors and is given by

\[ \delta(\mu_i, \mu_j) = \frac{1}{D} \sum_{k=1}^{D} \left[ w(k) (\mu_i(k) - \mu_j(k)) \right]^2, \]  

where \( \mu_i \) and \( \mu_j \) are the mean vectors of the \( i^{th} \) and the \( j^{th} \) mixture components in the model set, \( D \) is the dimension of the observation, \( w(k) \) is the weight and \( \mu_k(k) \) is the \( k^{th} \) component of the mean vector \( \mu_i \).

The mixture components are clustered into \( \Phi \) clusters. The center of the cluster \( \xi_k \) is given by [6],

\[ c_k = \frac{1}{\text{size}(\xi_k)} \sum_{n \in \xi_k} \mu_n, \quad \phi = 1, \ldots, \Phi, \]  

where, \( \Phi \) is the total number of clusters and \( \mu_n \) is the mean vector of the \( n^{th} \) mixture component associated with the cluster \( \xi_k \).

Each mixture component is associated with a set of clusters that are close to it. The selection is based upon

\[ \chi(m) = \delta(c_{\phi}, \mu_m) \quad \text{for} \quad m = 1 \ldots M, \]  

where \( M \) is the number of mixtures in the state. The closest \( N \) mixtures are then associated with the cluster \( \xi_k \). This process is repeated for all clusters \( \xi_k \) for \( \phi = 1, \ldots, \Phi \), and for each state of every HMM.

During recognition, the cluster that is closest to the observation is selected using the criterion

\[ \hat{\theta}_i = \arg \min_{\phi = 1, \Phi} \delta(c_{\phi}, \alpha_i), \]  

where, \( \alpha_i \) is the current observation vector, and \( \hat{\theta}_i \) is the index of the cluster selected for this observation. Only mixtures that are associated with this cluster are used for likelihood computation. This reduces the computational requirement on likelihood computation, yet doesn’t significantly sacrifice recognition accuracy.

In our experiments below, we cluster mixture components into 16 VQ clusters. Table 8 compares the performance of VQ-based recognition with the fixed-point non-VQ-based recognition. When each cluster is associated with 8 mixtures of every state, there is almost a 50% reduction in CPU usage for state-likelihood computation. Table 8 reflects this with a 30.4% reduction in CPU usage. Despite the significant reduction in complexity, there is little degradation in performance. In the case when each cluster is associated with 4 mixtures of every state, the performance is fairly degraded. Figure 2 compares the number of mixture computation per frame, of the non VQ-based decoder with the VQ-based decoder for a particular utterance. We observe that the peak CPU usage is reduced by almost 50%.

7 RESULT ANALYSIS
The use of 2-Level CMS leads to a significant improvement in performance. A 70% reduction in digit error is observed as compared to the no-CMS case. Further improvement in performance can be seen by using discriminative training.

An analysis of substitution errors shows some interesting results. Table 9 tracks the top 5 substitution errors in
<table>
<thead>
<tr>
<th>VQ Size</th>
<th>% CPU</th>
<th>Digit Error</th>
<th>Ins. Error</th>
<th>Del. Error</th>
<th>Subst. Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>49.2</td>
<td>4.6%</td>
<td>0.2%</td>
<td>2.0%</td>
<td>2.4%</td>
</tr>
<tr>
<td>8</td>
<td>34.2</td>
<td>5.2%</td>
<td>0.4%</td>
<td>2.2%</td>
<td>2.6%</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>5.7%</td>
<td>0.4%</td>
<td>2.2%</td>
<td>3.1%</td>
</tr>
</tbody>
</table>

Table 8. Performance degradation due to VQ based mixture selection.

![Figure 2](image)

Figure 2. Mixtures computed for the non-VQ based and VQ-based decoder, for the digit sequence “one zero six” at a beam width of 150.

The experiments that perform algorithmic improvements. The substitution of 5 → 9 occurs more frequently than the substitution of 9 → 5. A possible explanation is that the /r/ sound in 5 is dropped quite often. The /r/ sound of 5 is identical to the /l/ sound of 9. So when the /r/ is dropped, the recognizer has to discriminate between 5 and 9 solely on the /r/ versus /l/. Discrimination. The reverse substitution is less likely since the ending /n/ sound is usually not dropped. We notice that discriminative training offers some improvement to correct this problem.

The substitution of the digit 8 → 3, seems to occur when the digit 4 precedes 8. In those cases 4 tends to be deleted, i.e., 4 and 8 are merged into 3. The ending /r/ sound of 4 is usually soft and the /oh/ sound of 4 glides to the beginning /ay/ sound of 8. The substitution of 8 → 2, occurs sometimes when 8 precedes 8. The /t/ sound of the first 8 is fairly strong while the same sound is very soft for the second 8.

8 SUMMARY

In this paper, we have described a real-time system for connected-digit recognition in a low-SNR mobile environment such as a car. We have shown algorithms that achieve improved recognition performance without significant increase in computational complexity. Figure 8 shows the performance comparison of the various algorithms presented. We also described the performance of a real-time system that uses the Lucent’s DSP1637 fixed-point digital signal processor and algorithms for complexity reduction.

Currently we are investigating low-complexity methods of obtaining higher performance including optimal feature selection, a 3-state background model for absorbing extraneous sounds, and training using real car data.

9 ACKNOWLEDGEMENTS

The authors would like to thank Greg Ciurpita for his assistance in the experiments with the fixed-point DSP system and the computation of CPU usage presented in section 6.2.

REFERENCES