High-Performance Low-Latency Speech Recognition Via Multi-Layered Feature Streaming and Fast Gaussian Computation

Liang Gu, Jian Xue, Xiaodong Cui and Yuqing Gao

IBM T. J. Watson Research Center, Yorktown Heights, NY 10598

Abstract

Highly accurate speech recognition with very low latency is a big challenge but also an important requirement for modern real-time speech recognition applications such as speech-to-speech translation. We attack this problem by proposing a highly effective and efficient streaming mode decoding scheme. A novel multi-layered feature streaming method is introduced to minimize truncation errors during streaming by optimizing look-ahead parameters. A set of speed-up algorithms are further proposed to speed up both Gaussian computation and graph search. Experiments show dramatic reduction in decoding latency using the proposed decoding scheme, with high recognition accuracy similar to utterance based decoding.

1. Introduction

Automatic speech recognition has been a long-time grand challenge due to significant variations in speaker accents, speaking styles and background noise conditions. In recent years, automatic speech recognition performance has been greatly improved by 1) training speech models using a large amount of speech and language data, and 2) applying a number of advanced and effective algorithms such as MPE training [1], fMPE front-end analysis [2], static graph decoding [3], etc. While these approaches achieve superior recognition accuracy, they also increase significantly the complexity of the speech decoding procedures, and hence lead to much slower decoding speed when compared with conventional speech recognizers that were trained on much smaller data sets and using simpler algorithms. This decoding speed issue is now a big concern for the design of most modern real-time speech recognition applications such as speech-to-speech translation where computational power is limited.

Another important performance factor related to decoding speed is decoding latency, which is the overall computational processing time spent after the whole speech utterance is received. In real-time speech recognition applications, the decoding latency not only depends on the decoding speed, but also depends on the style of the decoding procedure. In particular, there are three common decoding styles based on different flavors of the Viterbi algorithm [4]: two-pass utterance based decoding, one-pass utterance based decoding and one-pass streaming mode decoding. The recognition accuracy can be enhanced via two-pass and one-pass utterance based decoding because the contextual information within each utterance and between consecutive utterances may be exploited, at the cost of larger decoding latency. On the other hand, one-pass streaming mode decoding aims at very low decoding latency by limiting the length of look-ahead and usually requires the back-tracking function to be truncated to a certain number of frames, and may therefore result in lower recognition accuracy.

In this paper, we propose a new method of high performance speech recognition that achieves high recognition accuracy and low decoding latency based on streaming mode decoding. Particularly, a novel multi-layered feature streaming method is proposed to minimize both the truncation error during streaming mode decoding and the computation overhead during feature look-ahead. An effective cepstral mean normalization (CMN) algorithm is further designed for streaming mode decoding. A fast Gaussian computation method is applied that maximizes the posterior increase estimations [5]. Additionally, a set of novel speed-up schemes are introduced and implemented including class-adjusted fast Gaussian selection, class-based dynamic beam pruning, parallel Gaussian computation on multi-core CPUs, etc.

Our baseline system is an IBM state-of-the-art speech recognition engine designed under DARPA Effective Reusable Speech-to-Text (EARS) [4] and Global Autonomous Language Exploitation (GALE) programs [6]. The baseline recognizer contains a number of the most advanced speech recognition components including model space Minimum Phone Error (MPE) training [1], feature space MPE (fMPE) training [2], static graph search [3], etc.

Our proposed new algorithms were applied on the above baseline system for the DARPA the TransTac Speech-to-Speech Translation program, which requires both high recognition accuracy and low decoding latency. The goal of our work in this paper is to maintain the recognition accuracy of our best one-pass utterance based recognizer while reducing speech recognition latency to less than 100 ms so that the end-to-end speech-to-speech translation system can be performed in real-time and with very low latency.

2. Multi-Layered Feature Streaming

A. Utterance based Speech Decoding

The process of speech decoding in automatic speech recognition could be stated as follows: Find the most likely word sequence \( W = \{w_1, w_2, \ldots, w_N\} \), given a sequence of acoustic observations \( X = \{x_1, x_2, \ldots, x_T\} \). Typically, this is accomplished by assuming a probabilistic model of speech production where \( X \) is produced by \( W \) with probability \( P(W|X) \). Using Bayes’ Rule, the best word hypothesis with the maximum a posteriori (MAP) probability given the feature sets \( X \) of a whole utterance can be formulated as:

\[
\hat{W} = \arg \max_w P(W|X) = \arg \max_w P(X|W)p(W),
\]

where \( P(X|W) \) and \( p(W) \) are the acoustic model and language model probabilities, respectively.
In one-pass utterance based decoding, the feature sets are extracted from the whole speech utterance. In the IBM system, the ultimate feature sets are computed through multiple feature layers. A final feature set \( \mathbf{X} \) representing an utterance with input waveform set \( \mathbf{Y} = \{y_1, y_2, \ldots, y_T\} \) can be depicted as:

\[
\mathbf{X}^f_1 = \Phi(\mathbf{y}^f_1) = \Phi_{\text{MPE}} \{ \Phi_{\text{MLLR}} \{ \Phi_{\text{LDA}} \{ \Phi_{\text{CMN}} \{ \Phi_{\text{MFCC}} \{ \mathbf{Y}_1 \} \} \} \} \}
\]

(2)

where IMPE refers to feature space MPE, fMLLR refers to feature space maximum likelihood linear regression (MLLR), LDA refers to linear discriminant analysis, CMN refers to cepstral mean normalization, and MFCC refers to mel-frequency cepstral coefficients.

**B. Streaming mode decoding**

In streaming mode decoding, feature set \( \mathbf{X} \) is computed incrementally in the time domain. Compared with equation (2), the final feature set \( \mathbf{X}^f_1 \) is computed as:

\[
\mathbf{Y}_{\text{MFCC}}^f_1 = \Phi_{\text{MFCC}} \{ \mathbf{y}^f_1 \}
\]

(3)

\[
\mathbf{Y}_{\text{CMN}}^f_1 = \Phi_{\text{CMN}} \{ \mathbf{Y}_{\text{MFCC}}^f_1 \}
\]

(4)

\[
\mathbf{Y}_{\text{LDA}}^f_1 = \Phi_{\text{LDA}} \{ \mathbf{Y}_{\text{CMN}}^f_1 \}
\]

(5)

\[
\mathbf{Y}_{\text{fMLLR}}^f_1 = \Phi_{\text{fMLLR}} \{ \mathbf{Y}_{\text{LDA}}^f_1 \}
\]

(6)

\[
\mathbf{X}^f_1 = \Phi_{\text{fMPE}} \{ \mathbf{Y}_{\text{fMLLR}}^f_1 \}
\]

(7)

where \( \mathbf{W}_{\text{MFCC}}, \mathbf{W}_{\text{CMN}}, \mathbf{W}_{\text{LDA}}, \mathbf{W}_{\text{fMLLR}} \) and \( \mathbf{W}_{\text{fMPE}} \) are feature-layer dependent look-ahead decoding parameters.

As a result, equation (7) may be re-written as

\[
\mathbf{X}^f_1 = \Phi(\mathbf{y}^f_1) = \Phi_{\text{fMPE}} \{ \mathbf{Y}_{\text{fMLLR}}^f_1 \}
\]

(8)

We may optimize these look-ahead parameters by maximizing the overall likelihood of acoustic training data as

\[
\mathbf{Y}_{\text{fMLLR}}^f_1 = \arg \max_{\mathbf{Y}_{\text{fMLLR}}^f_1} \sum \log P(\mathbf{X}^f_1 | \mathbf{Y}_{\text{fMLLR}}^f_1)
\]

(9)

Since there are only a few parameters, we can also tune these parameters on a development set, as described in section 4.

**C. Cepstral mean normalization in streaming mode decoding**

In our streaming mode decoding, cepstral mean normalization (CMN) [7] is performed chunk by chunk. Let us assume the chunk size is \( T_c \), we propose computing CMN features as:

\[
\mathbf{Y}_{\text{CMN}}^{f_1-T_c} = \Phi_{\text{CMN}} \{ \mathbf{Y}_{\text{MFCC}}^{f_1-T_c} \} = \mathbf{Y}_{\text{MFCC}}^{f_1-T_c} - \mathbf{r}_1^{f_1-T_c}, \quad \mathbf{r}_1^{f_1-T_c} = \left( \lambda \mathbf{r}_1^{f_1-T_c} + T_c \right) / (\lambda + T_c)
\]

(10)

\[
\lambda = \frac{1}{\left( \mathbf{r}_1^{f_1-T_c} \right)^T \mathbf{r}_1^{f_1-T_c} / T_c}
\]

(11)

where \( \mathbf{r}_1^{f_1-T_c} \) and \( \mathbf{r}_1^{f_1-T_c} \) are the cumulative and short-term cepstral means, respectively. \( \lambda \) is the interpolation parameter that accommodates both the long-term and short-term speech feature characteristics.

**D. Feature space MLLR (fMLLR)**

We implemented fMLLR [8] for fast speaker adaptation as

\[
\mathbf{Y}_{\text{fMLLR}} = \mathbf{A} \mathbf{Y}_{\text{fMPE}} + \mathbf{b}
\]

(12)

where \( \mathbf{A} \) is the fMLLR linear transform on the feature space and \( \mathbf{b} \) is the bias term. In streaming mode decoding, \( \mathbf{W}_{\text{fMLLR}} \) depends on the amount of acoustic features collected and \( \mathbf{A} \) is updated only at utterance or sentence boundaries.

**E. fMPE streaming**

fMPE front-end analysis in equation (7) performs discriminative training in the feature space as

\[
\mathbf{X}^f_1 = \mathbf{Y}_{\text{fMLLR}}^{f_1} + \mathbf{M} \cdot \mathbf{h}_t
\]

(13)

where the original feature \( \mathbf{Y}_{\text{fMLLR}} \) is transformed into a very high (but sparse) dimensional space in \( \mathbf{h}_t \) and the projection matrix \( \mathbf{M} \) is trained under MPE criterion.

To form \( \mathbf{h}_t \), the posterior probabilities of current frame, \( \gamma_t' \), against a collection of Gaussians and its offset from each Gaussian

\[
\mathbf{P}_t = \{p_{\gamma_1'}, \gamma_1' \sigma_1', p_{\gamma_2'}, \gamma_2' \sigma_2', \ldots, p_{\gamma_N'}, \gamma_N' \sigma_N' \}
\]

are computed, where \( d \) is the feature dimension. Afterwards, acoustic contexts of the current frame are taken into account by averaging over adjacent frames and splicing them. For instance,

\[
\hat{h}_t = [p_{\gamma_1'}, \gamma_1' \sigma_1', p_{\gamma_2'}, \gamma_2' \sigma_2', \ldots, p_{\gamma_N'}, \gamma_N' \sigma_N', p_{\gamma_{N+1}}, \gamma_{N+1} \sigma_{N+1}, p_{\gamma_{N+2}}, \gamma_{N+2} \sigma_{N+2}, \ldots, p_{\gamma_{2N}}, \gamma_{2N} \sigma_{2N}]
\]

for \([-6, +6]\) neighborhood of the current frame. The number of neighboring frames affects both accuracy and speed. This range is normally set as \([-40, 40]\) to cover a wide span of acoustic contexts. In our streaming mode decoding, the range is reduced to \([-10, 10]\) to significantly reduce the latency while minimizing the accuracy degradation compared with utterance based decoding.

3. Fast Gaussian Computation

**A. Maximum Probability Increase Estimation**

In streaming mode decoding, the average decoding time must be less than 1x real-time to prevent exponential increase of recognition response time due to accumulation of decoding latency for each processing chunk. In our IBM baseline system, up to 80% of the decoding time is spent on Gaussian computation during acoustic modeling and fMPE front-end analysis. In order to speedup the Gaussian computation process and thereby the average decoding time, we proposed and implemented a set of new algorithms, as described next.

In statistical modeling for speech recognition, a single Gaussian distribution may be denoted as:
\[
p_{s}(x) = N(x; \mu_{s}, \Sigma_{s}) = \frac{1}{\sqrt{(2\pi)^{d}|\Sigma_{s}|}} \exp\left(-\frac{1}{2}(x - \mu_{s})^T \Sigma_{s}^{-1}(x - \mu_{s})\right)
\]

where \(\mu_{s}\) and \(\Sigma_{s}\) are the mean vector and covariance matrix for Gaussian distribution \(m_{s}\), respectively, and \(x\) is a speech feature vector with dimension \(D\) defined in equation (7).

To reduce the total number of explicit Gaussian likelihood computation, a \textit{Maximum Probability Increase Estimation} (MPIE) method is proposed and implemented that exploits the Gaussian likelihood scores of previous speech frames to speedup likelihood computation for the current frame. When diagonal covariance matrices are used, the logarithmic Gaussian likelihood can be obtained as:

\[
\log p_{\mu}(x) = c_{\mu} - 0.5 \sum_{j=1}^{D} (x_{j} - \mu_{\mu,j})^{2} \nu_{\mu,j},
\]

where \(c_{\mu}\) is a constant for each Gaussian and \(\nu_{\mu,j}\) are the diagonal elements of the inverse covariance matrix. Given two observations \(o_{t}\) and \(o_{t'}\) from frames \(t\) and \(t'\), respectively, we may write an equivalent equation:

\[
\log p_{\mu}(x_{t}) = \log p_{\mu}(x_{t'}) + \sum_{d=1}^{D} \left[ A_{d} + B_{d} \cdot \nu_{\mu,d} \right] \cdot \nu_{\mu,d},
\]

where \(A_{d} = 0.5 \cdot (x_{t,d} - x_{t',d})^{2}\) and \(B_{d} = x_{t,d} - x_{t',d}\).

Vectors \(A\) and \(B\) are constant for all Gaussian mixtures and the corresponding computational cost is hence negligible.

In MPIE, we estimate an upper bound of the second term in equation (15) as:

\[
U = \sum_{d=1}^{D} U_{d} = \sum_{d=1}^{D} \left[ A_{d} + B_{d} \cdot \nu_{\mu,d} \right] \cdot \nu_{\mu,d},
\]

where \(U_{d}\) is the maximum possible probability improvement for dimension \(d\). As a result, we get an upper bound of the logarithmic likelihood for frame \(t\) as:

\[
\log p_{\mu}(x_{t}) \leq \log p_{\mu}(x_{t'}) + U.
\]

In this way, the number of Gaussian likelihood requiring explicit computation is dramatically reduced and therefore leads to much less average decoding time with same recognition accuracy.

**B. Class-adjusted Fast Gaussian Selection**

In an HMM, the emission density function of a state \(s\) is typically parameterized by a mixture of Gaussian densities. The state conditional likelihood of a given observation vector \(x\) at time \(t\) may be denoted as:

\[
p(x \mid s) = \sum_{g \in G(s)} p(g \mid s) p(x \mid g),
\]

where \(G(s)\) is the set of Gaussian densities that makes up the Gaussian mixture distributions for state \(s\).

Note that, at a given frame, only a small subset of Gaussian components in the total Gaussian pool is significant to the likelihood computation. A cluster-based fast Gaussian selection method [4] can be used to decide which Gaussian components will contribute to the likelihood computation. We clustered all the Gaussian components in the total Gaussian pool into certain number of Gaussian clusters, and generated a Gaussian kernel for each cluster. In decoding search, for each observation vector \(x\), we firstly calculate the likelihood on the Gaussian kernels and get top \(N\) clusters based on ranked likelihood scores. The Gaussian components belonging to the top \(N\) clusters comprise the active Gaussian pool \(Y\). Based on \(Y\), the conditional likelihood of a state is computed using the following approximation:

\[
p(x \mid s) = \sum_{g \in Y} p(g \mid s) p(x \mid g)
\]

In our system we dynamically adjust the value of \(N\) based on speech/non-speech classification. If the speech frame is speech, the value of \(N\) is set to \(N_{s}\), otherwise \(N_{n}\). \(N_{s}\) is much smaller than \(N_{n}\), since for non-speech audio, we do not need to calculate a lot of Gaussian components. Such reduction in Gaussian computation will speedup decoding time significantly.

**C. Class-based dynamical beam width**

Beam pruning is used to retain only hypotheses with a score close to the best hypothesis for further consideration. Denoting the best scoring hypothesis by \(Q_{AC}(t) = \max_{p} Q(t, p)\)

\[
Q(t, p) = Q(t, p)
\]

We prune a hypothesis \((t, p)\) if

\[
Q(t, p) < f_{AC} \cdot Q_{AC}(t)
\]

Decoding noise is much slower than decoding speech due to higher confusion involved in the former procedure. Therefore, we set the beam width \(f_{AC}\) based on speech/non-speech detection. If current frame is speech, the value of \(f_{AC}\) is set to \(f_{s}\), otherwise to \(f_{n}\). In practice, \(f_{s}\) is much smaller than \(f_{n}\).

**D. Multi-thread computation on Multi-core CPUs**

Speech decoding time can also be reduced by taking advantage of unique hardware computational capabilities. A typical example is the parallel computational processing power provided by modern multi-core CPUs. In order to utilize the computational resources of multiple processing cores for a single streaming mode speech recognizer, we implemented multi-threaded computation for both Gaussian computation and Viterbi search. At each frame, the search space is distributed among threads by dividing active states into several parts, and assigning different parts to different threads. Before search begins, the likelihood of all the Gaussian kernels are pre-computed to generate the active Gaussian pool \(Y\). Then the partial active paths are extended by different threads.

**E. Programming via embedded assembly code**

Advanced computer programming is a straight-forward way to improve speech decoding speed without losing recognition accuracy. One approach we proposed and implemented in our system is to perform multiple-dimension Gaussian computation using embedded assembly code. Since we used diagonal covariance matrix in the Gaussian density function, we computed the likelihood of four dimensions at the same time. Such coding optimizations could increase the overall
decoding speed by about 30%, as discussed in the next section.

4. Experiments

A. Streaming mode vs. Utterance mode

Our proposed algorithms for high-performance streaming mode decoding were evaluated on English and Iraqi Arabic large vocabulary spontaneous speech recognition test sets collected in the DARPA TransTac speech-to-speech translation program. There are two categories of test sets, “lab” and “field,” which contain TransTac July 2007 evaluation data recorded during conversation in laboratory and field conditions, respectively. Each set has about 400 utterances. Only male speakers were evaluated in English and both male and female speakers were evaluated in Iraqi Arabic.

Speech features defined in equations (3)-(7) were used. The language model is a word-based interpolated language model. The English recognition model contains 55K Gaussian distributions and covers 50K unique words. The Iraqi Arabic recognition model contains 100K Gaussian distributions and covers 100K unique words. A finite state transducer is pre-compiled for the static graph decoder described in [4].

We first carried out experiments to compare recognition accuracy using utterance based decoding and streaming based decoding. As we are trying to perform recognition during face-to-face communications, two-pass utterance based decoding is not feasible and therefore only one-pass utterance based decoding is set as the baseline decoding method. The experimental results are summarized in table 1. In these experiments, we tuned the streaming mode lookahead parameters in equation (8) on the development sets and set \( \{w_{l1},w_{l2},w_{l3},w_{l4}\} \) as \( \{10,50,90,12000,100\} \), where the unit is milliseconds. The beam factors for pruning were set to maximize recognition accuracy (13 for English and 12 for Iraqi Arabic, respectively).

Table 1 shows that our proposed streaming mode decoding achieved almost the same recognition accuracy as one-pass utterance based decoding. This is because the multi-layered feature streaming minimized the distortion during the boundaries of speech chunks (25ms in these experiments). CMN and fMLLR are also carefully designed (as described in sections 2.C and 2.D) to minimize the performance loss due to the lack of contextual information in streaming mode compared with utterance based decoding.

B. Low-latency decoding by speedup algorithms

The speed of high-performance streaming decoding can be increased using algorithms introduced in section 3. To achieve very high recognition accuracy for all the users including people with strong accents, we managed to maintain the best system performance by setting a high beam factor value for Viterbi search pruning and a high a large number of level one prototypes for the Gaussian selection procedure. Based on this, we reduced the overall decoding time from 1.4x real-time to about 0.3x real-time and the recognition latency from 5 seconds to about 50 milliseconds on a MS Windows based laptop with Intel 2.0GHz Dual-Core CPU. The experimental results are listed in Table 2. It shows that the decoding latency can be reduced drastically if the overall decoding time can be managed within 1x real-time and hence the potential exponential accumulation of decoding delay is avoided.

5. Conclusion

Speech recognition accuracy and speech decoding latency are two critical performance factors for modern real-time speech recognition applications such as speech-to-speech recognition. We propose a new streaming mode decoding methodology via multi-layered feature extraction to minimize the accuracy gap between streaming mode decoding and utterance based decoding. This streaming mode decoder is further optimized in decoding time and decoding latency by using a set of novel speed-up algorithms including maximum posteriori increase estimation, class-based setting of decoding parameters, and parallel Gaussian computation on multi-CPUs. In our experiments, the proposed speech decoder achieved less than 100 ms average latency and almost the same recognition accuracy compared with our best utterance based speech recognizer.

6. References