Four-Layer Categorization Scheme of Fast GMM Computation Techniques in Large Vocabulary Continuous Speech Recognition Systems

Arthur Chan, Jahanzeb Sherwani, Ravishankar Mosur, Alex Rudnicky

Computer Science Department
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA, 15213.
{archan,jsherwan,rkm,air}@cs.cmu.edu

Abstract

Large vocabulary continuous speech recognition systems are known to be computationally intensive. A major bottleneck is the Gaussian mixture model (GMM) computation and various techniques have been proposed to address this problem. We present a systematic study of fast GMM computation techniques. As there are a large number of these and it is impractical to exhaustively evaluate all of them, we first categorized techniques into four layers and selected representative ones to evaluate in each layer. Based on this framework of study, we provide a detailed analysis and comparison of GMM computation techniques from the four-layer perspective and explore two subtle practical issues, 1) how different techniques can be combined effectively and 2) how beam pruning will affect the performance of GMM computation techniques. All techniques are evaluated in the CMU Communicator domain. We also compare their performance with others reported in the literature.

1. INTRODUCTION

The Gaussian mixture model (GMM) computation component of large vocabulary continuous speech recognition (LVCSR) systems is well known to be computationally intensive and a bottleneck in recognition. In a typical hidden Markov model-based system, the number of tied-triphone states (or senones) can be 2000 to 6000 each of which is a weighted sum of multi-dimensional Gaussian distributions. The GMM computation task involves the computation of the likelihood for all these tied-triphone states. In some tasks, GMM computation can consume 70-80% of the whole decoding process. As a result, much effort has been spent on optimizing GMM computation ([2]-[8]). Many techniques focus on approximating the full computation of all GMMs. For example, Gaussian Selection (such as [6]) assumes that only a few Gaussians will dominate the score of a GMM. This set of techniques has been found to be highly useful in practice. [1]

In this paper, we present a systematic study of fast GMM computation techniques. Our study is motivated by our desire to build highly-responsive conversational agents based on highly-accuracy, user-adaptive and real-time speech recognition. Our approach has been to refine an existing decoder, Sphinx 3.3 [2], to incorporate efficient GMM computation techniques. Our target is to achieve real-time performance with minimal accuracy degradation ($< 5\%$ relative) from Sphinx 3.3. The evaluation in this paper will focus on this set of techniques.

[1] Many researchers note the importance of machine optimization; however, this paper will focus on algorithmic optimization.

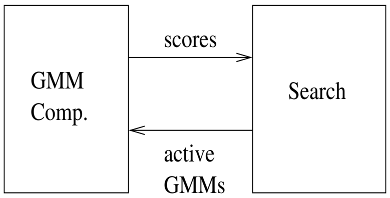

Figure 1: *The conceptual model of computation at every frame of a speech recognizer*

The conceptual model of major computation process of Sphinx 3.3, as well as many other HMM-based speech recognizers, can be illustrated in Figure 1. At every frame, the GMM computation module computes the scores for each active clustered GMM and feeds them into the search module. The search module traverses the word lattice and determines the highest scored paths. The number of paths traversed is controlled by the use of pruning techniques. This conceptual model is advantageous because it simplifies the discussion of different speed-up techniques and enhances interpretation by breaking down the measurement into two parts.

We further break down the GMM computation component into four different layers, with lower layers providing information to upper layers. Individual fast GMM computation techniques can be associated with specific layers. This property of the four-layer model allows us to select representative techniques from each layer and to compare the effectiveness of techniques from each layer. In this we follow Takami et al. [9], where tying techniques are similarly organized into four independent levels. We will further explore two subtle but significant issues of GMM computation techniques. The first is how the different techniques can be combined. The second is how pruning affects the performance of GMM computation techniques. We will try to analyze these issues from the four-layer perspective.

The organization of this paper is as follows, we first describe the detail of our four-layer categorization schemes for fast GMM computation techniques in Section 2. We will describe the details of each layer. The discussion of potential effect of pruning on these techniques under the four-layer categorization perspective can be found in Section 3. The evaluation conditions and results of representative techniques can be found in Section 4. The conclusion and future work can be found in Section 5.

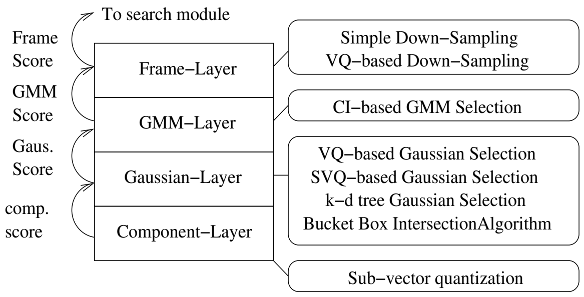

Figure 2: *Illustrated Four-Layer Categorization Scheme.*

2. FOUR-LAYER CATEGORIZATION SCHEME FOR FAST GMM COMPUTATION

We will describe the proposed categorization scheme in this section and the scope of our study.

2.1. FRAME-LAYER ALGORITHMS

Algorithms that decide whether a frame's GMM scores should be computed or skipped are categorized as frame-layer algorithms. In our discussion, we will assume the score of a skipped frame will be copied from the most recently computed frame.[2] The simplest example is to compute frame scores only every other frame, which we call this simple down-sampling (SDS). It is evaluated in [4].

Other than SDS, we also evaluate another scheme in which a VQ codebook is trained from all means of GMMs of a set of trained acoustic models. Then, in decoding, every frame's feature vector is quantized using that codebook. A frame is skipped only if its feature vector was quantized to a codeword which is the same as that of the previous frame. We call this method VQ-based Down-Sampling (VQDS) and it is slightly different from the work in [4] because we made use of acoustic modeling information in the algorithm.

2.2. GMM-LAYER ALGORITHMS

Algorithms that ignore some GMMs in the computation in each computed frame are assigned to the GMM-layer. One representative technique is context-independent (CI) GMM-based GMM selection (CIGMMS) proposed by Lee et al. in Julius [5]. Briefly the algorithm can be implemented as follows.[3]. At every frame, CI GMMs' scores are first computed and a beam is applied to these scores. For all context-dependent (CD) GMMs, if the corresponding CI GMM's score is within the beam, compute the detail CD GMM's score. If not, the CD GMM's score is backed-off by the corresponding CI GMM's score.

This scheme is highly effective in reducing GMM computation. However, because some scores are backed-off, they become the same when they are fed into the search module. As a consequence, beam pruning becomes less effective. We will explore this further in Section 3.

2.3. GAUSSIAN-LAYER ALGORITHMS

Usually only a few Gaussians will dominate the likelihood of a GMM and different techniques are used to decide which Gaussian dominates the likelihood computation. We categorize these techniques as part of the Gaussian layer. Generally a rough model, either vector-quantizer (VQ) [6], sub-vector-quantizer(SVQ) [2] or kd-tree [4] is first used to decide which Gaussian should be computed in the GMM. Hence, these techniques are also called Gaussian selection.

[2]This implementation can preserve transition information.

[3]This is slightly different from [5] which use fix number instead of a beam.

The major issue in using any Gaussian selection techniques is the need to trade-off between rough model computation (e.g. the VQ codebook) and accuracy degradation. Usually, a more detailed model (e.g., a higher-ordered VQ codebook) gives better accuracy, but results in more computation.

In this study, we focused on two techniques which made use of simple VQ as a Gaussian selector (VQGS) [6] and SVQ as a Gaussian selector (SVQGS) [2]. We ignored tree-based techniques because of the practical difficulty in then applying adaptation techniques such as Maximum Likelihood Linear Regression (MLLR).

2.4. COMPONENT-LAYER ALGORITHMS

As in the full-feature space, the distribution of features in a projection of the full-space (or subspace) can be approximated as a GMM. The full-space likelihood can thus be obtained by summing individual sub-spaces likelihood. Similar to situation in the full-space, only few Gaussians will dominate the subspace likelihood in a particular subspace. Therefore, techniques can be used to choose the best Gaussians in individual subspaces and combined them. We categorize algorithms which make use of this fact to be component-layer algorithm. We will focus one representative technique in this layer, sub-vector quantization (SVQ) [8] in which VQ was used as a Gaussian selector in subspaces.

2.5. RELATIONS BETWEEN THE 4 LAYERS

Our categorization scheme results in a layered architecture for GMM computation and many fast GMM techniques can be categorized into one of the layers [4]. The advantage of this scheme is that one can follow a conceptual model when implementing different algorithms. It also simplifies the studies of interaction of different schemes. For example, once we understand that two techniques are in the same-layer (such as VQGS and SVQGS), we will probably not want to implement them in the same system, as that can only result in higher overhead.

3. PRUNING AND FAST GMM COMPUTATION ALGORITHMS

The performance of fast GMM computation techniques is usually less effective when a tighter beam is used in search. From the perspective of our conceptual model, pruning can be regarded as a GMM-layer algorithm, and as such, only affects the layers above, namely, the GMM and frame layers.

We summarize the effect below.

Relationship with Frame-Layer: We assume skipped frames' scores are copied from previous frames' scores. However, the search module will decide whether a clustered-GMM is active or not based on pruning. There will be problematic situations where some GMMs are active in the current frame but deactivated in previous frame. Hence, recomputation of active states is necessary. When the beam is tight, recomputation

[4]We also note that some techniques has the characteristics of multiple layers.

can be time-consuming and can cancel out the computation gain obtained from down-sampling.[5]

Relationship with GMM-Layer: As the GMM's scores will feed into the search module, one observation is that if CIGMMS applied, the search time increases. If the CI scores' beam (mentioned in Section 2.2) is tight, CIGMMS will cause many CD-GMMs' scores to be the same and also narrows the range of scores. Hence, the search module will be more computationally intensive. In practice, this problem can be solved by tightening the Viterbi beam.

4. EXPERIMENTAL RESULTS

GMM computation techniques were tested in the Communicator domain [1] using 85,735 utterances (about 60 hours of speech) to train the model. The model contains 2165 clustered-triphone states, where each state is represented by a GMM with 32 mixtures and each mixture contains 39 feature components (12 mel-frequency cepstral coefficients (MFCC) + Δ + $\Delta\Delta$ + energies of each). The vocabulary size is 2001 words and results are evaluated using the 2001 Communicator test set that contains 1691 utterances.

The baseline recognizer we used is Sphinx 3.3 [2] in which tree-structured lexicons are used along with language model look-ahead. For our purposes, we further refined the recognizer by implementing techniques in fast GMM computation and phoneme-lookahead pruning. Our implementation of the latter part can be viewed as a simplified version of work by Ortman et al. [3].[6]

We report the performance in terms of the word error rate (WER), computation required for GMM computation (GMM), search (Srch) and overhead (Ovrd) in terms of real-time factor (RTF). All results are evaluated on a 1GHz Intel Pentium III with 256MB RAM. The word error rate of the baseline recognizer is 18.65% and it takes 5.85xRT in GMM computation and 0.95xRT in search with un-tuned beam sizes.

The final choice of algorithms depends on both the speed and accuracy of the recognition. To ensure practical validity, we consider a technique or a combination of techniques usable only if they cause less than 5% relative degradation with respect to the baseline recognizer.

We will present the comparison of all techniques with un-tuned pruning beams in Section 4.1. The results of the combination of algorithms in the four layers will be described in Section 4.3. Finally, the effect of pruning will be discussed in Section 4.4.

4.1. COMPARISON OF REPRESENTATIVE TECHNIQUES IN THE FOUR-LAYER CATEGORIZATION WITHOUT PRUNING.

In this section, we evaluate the performance of different fast GMM algorithms. We followed the four-layer categorization scheme described in Section 2 and chose the following representative algorithms in each layer.

Frame-layer: 1) Simple down-sampling (SDS)[4], tuned with down-sampling ratio (D).

2) VQ-based down-sampling (VQDS) (proposed in Section2.1), tuned with different size of VQ codebook. (SZ)

[5]One can also deactivate the state. However, in our preliminary experiment, we found that deactivation can result in no valid paths at the final frame.

[6]This work is open-source but still under development. A development version can be found at `www.cs.cmu.edu/~archan`.

GMM-layer: CI-based GMM selection (CIGMMS) [5], tuned with different beam thresholds (CIT) applied to the CI GMM scores.

Gaussian-layer: 1) VQ-based Gaussian selection (VQGS) [6], tuned with different VQ codebook sizes (SZ).[7]

2) Sub-VQ-based Gaussian selection (SVQGS) [2], tuned with different VQ codebook sizes (SZ) when number of sub-vectors equal to 3.

Component-layer: Sub-vector quantization (SVQ) [8], we present results with different number of sub-vectors (NSV) when the codebook size is 4096.

These techniques have been selected because they are well-known and are representative of their class; most other published techniques are derivatives. The results are tabulated in Table 1.

Table 1: *Comparison of Fast GMM Computation Techniques*

Algorithms	WER	Total	GMM	Srch	Ovrd
BL	18.65	6.90	5.85	0.85	-
SDS($D = 2$)	19.10	4.35	3.39	0.96	-
VQDS($SZ = 2^{12}$)	18.89	6.27	5.08	0.97	0.22
CIGMMS($CIT = 10^4$)	**18.82**	**3.25**	**1.18**	**2.06**	**-**
VQGS($T = 1.5$)	18.95	3.95	2.84	0.89	0.22
SVQGS($SVQT = 10^{-4}$)	18.86	4.11	2.62	1.49	0.49
SVQ($NSV = 13$)	18.69	4.20	2.04	0.98	1.08

We note that CIGMMS gave us the best speed performance within the 5% degradation constraint (absolute WER below 19.55%). As mentioned in 3, there is a minor problem inherent in CIGMMS, where the range of GMM scores is narrowed. For example, at beam threshold 10^4, there is an 80% reduction in GMM computation but the search module requires 100% more time. Other techniques are less effective mainly because the overhead is too large. SVQ (with $NSV = 13$), for example, is highly ineffective in terms of overhead.

4.2. COMPARISON WITH PREVIOUSLY REPORTED RESULTS

Table 4 presents a comparison of fast GMM computation techniques from different sources. We found that some authors claim that the speed-up factor can be 9 (or 88.89% reduction); however, such techniques usually sacrifice about $> 10\%$ accuracy. Bearing in mind that different researchers have varying goals (speed, accuracy), we include only those results that have less than 5% degradation from the baseline used by the authors. We observe that 75%-80% time reduction appears to be an upper bound on GMM computation time savings for a system with 2k-6k tied states.

Table 2: *Relative reduction of GMM computation.*

Algorithms	GMM(Mix)	BL(RT)	Rel. Deg	RF.
SDS[4]	3k(32)	3	2	30%
CIGMMS[5][8]	2k(64)	NA	7.5%	72.09%
VQGS[6]	110(64)	NA	$< 5\%$	80%
VQGS[7]	NA	NA	$< 5\%$	77%
VQGS[10]	4K(16)	NA	$< 5\%$	51%
BBI[4][9]	3k(32)	2.2	5%	27%

When comparing different techniques appearing in the literature, it is important to be aware the use of the term "relative

[7]We also implemented the dual ring constraint described by [7]. We found that the improvement is negligible.

computation reduction". Relative computation reduction may be different when the system's complexity changes. For example, a technique is likely to give better results in a system with 64 mixtures than in one with 32 mixtures. The reason is that the former system may have more redundant parameters and can be more readily trimmed by fast computation techniques. Hence, it is very important to quote system performance in the context of specific system parameters. Likewise, it is important to report exact pruning conditions as this usually changes the baseline's complexity. Unfortunately, this is not a common practice.

4.3. ATTEMPTS IN COMBINING TECHNIQUES IN THE FOUR-LAYER CATEGORIZATION

From the results of Section 4.1, we tried to combine different algorithms using the best settings we obtained. Based on the best results we obtained so far, i.e. CIGMMS, we tried to improve its performance by augmenting it with other techniques. In this stage, we ignored less promising techniques such as VQDS and SVQ. The performance of the former was too ineffective and the latter had too much overhead. The results obtained from the combination of various techniques with CIGMMS can be found in Table 3.

In the last section we observed that most of the techniques in the literature provide less than 80% GMM computation reduction. This is confirmed by our experimental results. Empirically, we observed that only small gains can be obtained through augmenting CIGMMS with VQGS or SVQGS. The use of SDS did give us some gain; however, this sacrificed the performance to a point which was more than 5%. The fact that multiple techniques didn't give us much gain under the 5% constraint shows that 80% may be an empirical upper bound of the GMM computation reduction under the current complexity of the system when just using fast GMM techniques.

Table 3: *Results of techniques combined with CIGMMS($CIT = 10^4$)*

Algorithms	WER	Tot	GMM	Srch	Ovrd
BL	18.65	6.90	5.85	0.85	-
CIGMM	18.82	3.25	1.18	2.06	-
+SDS	19.76	3.02	0.53	2.47	-
+VQGS	18.57	3.23	1.37	1.63	0.22
+SVQGS	18.866	3.12	0.93	1.63	0.49

4.4. EFFECT OF PRUNING TECHNIQUES

The final set of results we provide illustrates the effect of pruning on GMM computation techniques. Consider that the overhead of VQGS and SVQGS can be very significant after pruning (even before pruning, the overheads are 0.22xRT and 0.47xRT respectively). Therefore, we decided to test with only CIGMMS applied, and incrementally added different beam-pruning techniques to improve performance. Table 4 summarizes these results.

The use of pruning can effectively solve the back-off problems of CIGMMS, though one might be surprised at the effectiveness of beam-pruning techniques compared to GMM techniques. One can probably understand this from the observation that the pruning threshold is continuous and can be more easily varied.

Table 4: *Beam applied to improve the search performance*

Algorithms	WER	Total	GMM	Srch
BL	18.65	6.90	5.85	0.85
CIGMMS	18.82	3.25	1.18	2.06
+Word-end	18.76	2.71	1.13	1.57
+Vit. Beam	19.06	1.63	0.91	0.71
+Phone Beam	19.12	1.49	0.90	0.59
+Word Beam	19.27	1.31	0.89	0.42
+Histogram	19.25	1.16	0.86	0.30
+P. Lookahead	19.49	1.11	0.82	0.28

5. CONCLUSION

We proposed a conceptual categorization scheme for fast GMM computation techniques. We compared different techniques under this scheme and then compared our best result with techniques that can be found in the literature. Our current test is limited to tasks which GMM computation dominates the decoding process's computation. In the future, we will evaluate in tasks where search is dominant.

6. ACKNOWLEDGEMENTS

This work, part of the CALO project, was supported by DARPA grant NBCH-D-03-0010. The content of the information in this publication does not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred. The authors would like to thank Richard Stern, Alan Black, Jim Baker and Satanjeev Banerjee for their valuable suggestions.

7. References

[1] Bennett, C. and Rudnicky, A.I. "The Carnegie Mellon Communicator Corpus", ICSLP 2002, Denver, Colorado, pp 341-344.

[2] Mosur, R., Singh, R., Raj B. and Stern, R.M. "The 1999 CMU 10X Real Time Broadcast News Transcription System", Maryland, 2000 Speech Transcription Workshop.

[3] Ortmanns, S., Ney, H., Coenen, N. and Eiden, A. "Lookahead Techniques for Fast Beam Search", ICASSP 1997, Munich, Germany.

[4] Woszczyna, M. "Fast Speaker Independent Large Vocabulary Continuous Speech Recognition", Universitat Karlsruhe; Institut fur Logik, Komplexitat und Deduktionssysteme. Dissertation. 1998.

[5] Lee, A. ""Gaussian Mixture Selection using Context-independent HMM," IEEE ICASSP, 2001.

[6] Bocchieri, E. "Vector quantization for efficient computation of continous density likelihoods", ICASSP, volume II, page II-692-II-695, Minneapolis, 1993.

[7] Gales, M. J. F., Knill, K. M. and Young, S. J. "Use of Gaussian Selection in Large Vocabulary Continuous Speech Recognition Using HMMs" ICSLP 1996. Philadelphia, Pennsylvania.

[8] Mosur R., Bisiani, R., and Thayer, E. "Sub-Vector Clustering to Improve Memory and Speed Performance of Acoustic Likelihood Computation" Eurospeech, Rhodes, Greece, Sep 1997.

[9] Satoshi, T. , Sagayama, S., "Four-Level Tied Structure for Efficient Representation of Acoustic Modeling" in ICASSP vol. 1 pp. 520-523, 1995

[10] Douglas P., "An Investigation of Gaussian Shortlists" ASRU 1999, Keystone, Colarado, USA.