SUBSPACE HIGH-DENSITY DISCRETE HIDDEN MARKOV MODEL

Guoli Ye, Brian Mak

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
{yeguoli, mak}@cse.ust.hk

ABSTRACT

Compared with continuous density hidden Markov model (CDHMM), discrete HMM (DHMM) has inherent attractive properties: it takes only $O(1)$ time to get the state output probability; and the discrete feature could be encoded in less bits compared with cepstral coefficients, which saves band width in distributed ASR architecture. Unfortunately, the recognition performance of conventional DHMM is significantly worse than that of CDHMM due to the large quantization error and the use of multiple independent streams. One way to reduce the quantization error, thus improving the recognition accuracy, is to use a very large codebook. However, the training data is usually not sufficient to robustly train such high density DHMM (HDDHMM). In this paper, we investigate a subspace approach together with subvector quantization to solve the training problem of HDDHMM. The resulting model is called subspace HDDHMM (SHDDHMM). On both Resource Management (RM) and Wall Street Journal (WSJ) 5K-vocabulary task, when compared with its CDHMM counterpart, SHDDHMM shows comparable performance in recognition accuracy, with faster decoding speed and lower bandwidth requirement.

***Index Terms*—** subspace modeling, subvector quantization, high-density discrete HMM, semi-continuous HMM

1. INTRODUCTION

For many current ASR systems, CDHMM with Gaussian mixture model (GMM) as state output distribution, is commonly adopted, due to its simple parametric form and good generalization ability. On the other hand, DHMM with discrete density as state output distribution, which was used in the early stage of speech recognition history, is almost abandoned nowadays, mainly because its recognition accuracy is not as good as that of CDHMM.

The unsatisfactory performance of traditional DHMM mainly arises from two reasons:

- Large quantization error: unlike GMM which has a smooth probability density function, discrete density imposes hard partitions in the feature space, with each partition represented by a codeword. All the different feature vectors falling into the same partition are quantized into the same codeword, which inevitably causes quantization error. Due to the limit of memory, and the amount of training data, traditional DHMM usually uses a codebook size of 256 to 1024. Such a small codebook size usually results in large quantization error, which badly hurts the recognition performance.
- Incorrect stream independent assumption: due to the use of large feature dimension (e.g., the common 39-dimensional MFCC vector) in ASR, multiple-stream DHMM is employed to allow manageable codebook sizes. The inability to model correlation among streams also limits the performance of DHMM.

With the advance of computer technology, we would like to revisit DHMM to see if some of its limitations could be overcome nowadays. Compared with CDHMM, DHMM is attractive to us for the following reasons:

- Fast decoding: in CDHMM, it is found that the Gaussian likelihood computation may take about 30-70% of the total recognition time [1]. What is worse, the computation time grows linearly with the number of Gaussian components. Large and complex systems usually use a large mixture of Gaussians per state, thus resulting in a heavy computation burden in decoding. In DHMM with each state storing a discrete density table, to get the state output probability only requires a $O(1)$ time table lookup, which is independent of the table size. [1]
- Bandwidth saving in distributed speech recognition (DSR) system: with the popularity of cloud computing technology, DSR systems become common. In these kind of systems, ASR components are distributed between the client and server. The speech feature is extracted locally at the client, and then transmitted to a remote server, where the recognition is carried out. The discrete feature used by DHMM could be encoded in much less number of bits than MFCC feature used in CDHMM, which makes it suitable to be transmitted in low bandwidth environment.

In recent years, there is the resurrection of using discrete systems to model speech [2, 3]. Unlike the unsupervised vector quantization methods (e.g., k-means clustering) commonly used in traditional DHMM, Droppo [2] proposed partitioning the feature space by an acoustic decision tree using MMI-based purity function at each tree node. By making using of the class label knowledge, which is not used in k-means clustering, the resulting discrete system showed comparable performance as that of GMM, but only on a simple frame classification task.

Teunen and Akamine [3] introduced mixture models and soft decisions to alleviate the quantization error of discrete density. Mixture

This work was supported by the Research Grants Council of the Hong Kong SAR under the grant numbers HKUST617008.

[1] Finding the codeword for an input vector in DHMM may also take time, depending on the vector quantization (VQ) method. e.g., if the simplest LBG method is used, finding the codeword takes $O(N)$ time for a codebook size of N; on the other hand, if tree VQ is used, it takes $O(\log N)$ time. Furthermore, for an input vector, the codeword only needs to be found once, and then used by all the active states.

models make use of the smoothing property of ensemble method, and soft decisions make soft partitions rather than hard partitions. However, these techniques incur additional computational cost. Furthermore, compared with CDHMM, there was still a 43% relative increase in word error rate (WER) of the proposed model on WSJ 5K task [3].

In [4], we proposed the *high-density discrete hidden Markov model* (HDDHMM) to reduce the quantization error of discrete density by simply increasing the codebook size. In theory, if the codebook size is infinitely large, then the quantization error will become infinitely small. In practice, limited by computational resources, we need to address the following three problems for HDDHMM:

- requirement of large memory to store the model
- how to find the codeword for an acoustic vector fast?
- lack of training data to train such a high-density model

The first two problems are not as serious as one may think. With the advance in semiconductor memory and its falling price, large memory requirement could be easily satisfied nowadays. To find the codeword for an acoustic vector fast, tree-structured, or subvector-quantized (SVQ) codebooks could be used instead of linear codebook. Unlike linear codebook in which the search time for a codeword grows linearly with the codebook size N, the tree-structured or SVQ codebooks only take $O(logN)$ time. In this paper, SVQ codebooks are used.

The lack of training data turns out to be a real challenge for HDDHMM with a codebook consisting of *tens of thousands* of codewords. For such a large codebook, even with today's large inventory of speech corpora, it is still hard to get sufficient training data for it. In [4], HDDHMM was directly converted from CDHMM without any retraining. In this paper, we investigate a subspace approach in which each state output discrete density is treated as a vector, and it is required to lie in the subspace spanned by a global pool of bases. The bases are further constrained with the use of SVQ in order to reduce the number of model parameters. Consequently, the number of model parameters in the resulting HDDHMM is greatly reduced and re-estimation of the model parameters becomes possible. We call our new HDDHMM "*subspace high-density discrete hidden Markov model*" (SHDDHMM).

2. SUBSPACE HIGH-DENSITY DISCRETE HMM

The overview structure of SHDDHMM is shown in Fig. 1. The details are explained in the following sections.

2.1. Subvector-quantized (SVQ) Codebooks

This section gives an introduction of SVQ codebooks, which are used by SHDDHMM. To construct the SVQ codebooks, each d-dimensional acoustic vector x_t is split into L subvectors[2], $x_t = [x_{1t}, x_{2t}, \ldots, x_{Lt}]$. Subvectors of each split are vector-quantized (by k-means) to create an SVQ codebook for the split. Let n_i denote the SVQ codebook size of the ith split.

Given a new acoustic vector, it will first be split into L subvectors. The subvector in the ith split is then compared with the corresponding SVQ codebook to find its SVQ codeword in $O(n_i)$ time. After getting all the L SVQ codewords of an input vector, its full-space VQ codeword is constructed as the products of SVQ codewords, one from each of the L splits as:

$$VQ(x_t) \equiv SVQ_1(x_{1t}) : SVQ_2(x_{2t}) : \cdots : SVQ_L(x_{Lt}),$$

[2]When $L = d$, the SVQ codebooks become SQ codebooks used in [4].

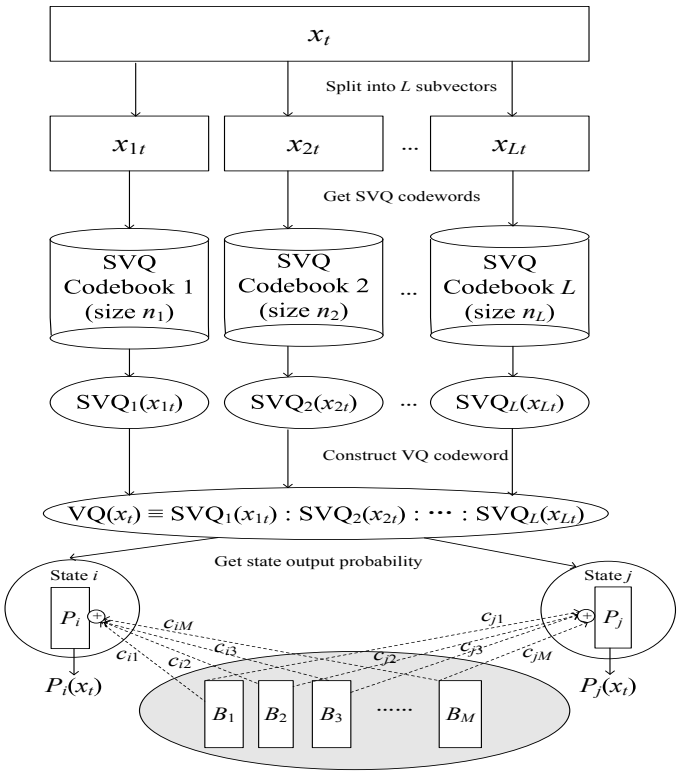

Fig. 1. SHDDHMM architecture overview. The shaded ellipse represents the global pool of bases, spanning the subspace. Each state output discrete density table (i.e., P_i, and P_j), lies in the subspace. The state dependent weights (the dotted lines) and the global pool of bases (the shaded ellipse) are temporary parameters, which only exist during the model training stage, while the final well-trained model only stores P_i and P_j.

where $VQ(\cdot)$ and $SVQ_i(\cdot)$ represent the VQ codeword given the full-space acoustic vector and the SVQ codeword in the ith split given the acoustic subvector in that split respectively. Notice that although SVQ is employed, it is only used to efficiently index a VQ codeword in the original full-space through the combinatorial effect of per-split SVQ codewords.

Since construction of full-space codeword from SVQ codewords takes no time, the time to get the full-space codeword for a given input is determined by the cost to find the L SVQ codewords, which is $(\sum_{i=1}^{L} n_i)$. Thus, by exploiting SVQ codebooks, finding codeword for an input vector is significantly quicker than that of the linear VQ codebook with the same codebook size, which takes $O(\prod_{i=1}^{L} n_i)$ time. Finally, the constructed full-space codeword is used as an index to get the state output probability by a table lookup, which takes only $O(1)$ time.

2.2. SHDDHMM with Constrained Bases

Like DHMM, SHDDHMM usually has multiple streams. For simplicity, we only discuss SHDDHMM with one stream in this section. It should be easy to extend it to multiple streams.

2.2.1. Basic Structure

The basic structure of SHDDHMM could be simply written as:

$$P_j(VQ(x_t)) = \sum_{m=1}^{M} c_{jm} B_m(VQ(x_t)) \quad (1)$$

where j is an HMM state; P_j is the output discrete density table of state j with $P_j(\cdot)$ being the discrete density function; $B_1, \cdots, B_M$ are a pool of M discrete density tables shared globally by all the states; $c_{j1}, \cdots, c_{jM}$ are the state-dependent weights, satisfying the constraint that $\sum_{m=1}^{M} c_{jm} = 1$. By viewing a discrete density table as a vector, the vector $P_j \in \Re^N$ thus lies in the subspace spanned by M bases $B_1, \cdots, B_M$ [3]. We call it "subspace" as the number of bases M in the model is usually significantly smaller than the dimension N of vector P_j, which is the discrete density table size or codebook size.

Since the codebook size N (i.e., the length of each basis vector) in SHDDHMM is usual very large (tens of thousands), even we use the subspace representation, there are still too many parameters in the system. Thus, similar as in [6], we made the assumption that: the SVQ codewords of different subvectors are conditionally independent given the basis. This assumption imposes a constraint on each basis as:

$$B_m(VQ(x_t)) = \prod_{i=1}^{L} B_{im}(SVQ_i(x_{it})) \quad (2)$$

where $B_{1m} \in \Re^{n_1}, \cdots, B_{Lm} \in \Re^{n_L}$ are the SVQ discrete density tables in split 1 to L respectively, which are used to construct the original basis $B_m \in \Re^N$. Thus, the "real" parameters in SHDDHMM with constrained bases are c_{jm}'s and B_{im}'s, which have significantly fewer number of parameters compared with the original system parameters P_j's.

Let us use an example to illustrate the reduction of system parameters in the proposed model. Let $I = 3132$ be the number of states in our tied-state system, $M = 1024$ be the number of bases, $L = 2$ so that a feature vector is split into 2 subvectors, and the SVQ codebook sizes for them are $n_1 = 256 = n_2 = 256$ respectively. Thus, the equivalent full-space codebook size, $N \equiv n1 * n2 = 65536$. In the original system, there are I states, each with a discrete table of size N, giving a total of $NI = 205,258,752$ parameters! On the other hand, in the proposed model, there will be only $MI = 3,207,168$ parameters for the weights c_{jm}'s, and $M(n_1 + n_2) = 524,288$ parameters for the constrained bases B_{im}'s, resulting in a total of 3,731,456 model parameters, which is 55-fold reduction compared with the original system.

Notice that the weight parameters c_{jm}'s and the basis parameters B_{im}'s only exist during the training stage, which make training with limited amount of data feasible. After training, we use these parameters to compute the state output discrete densities P_j's by Eqn.(1) and (2). Finally, we discard the weights and basis parameters, and only store the state output discrete densities. Thus, the final SHDDHMM for decoding has the same architecture of traditional DHMM, but just with a much larger codebook. In other words, the attractive properties of DHMM are naturally inherited by SHDDHMM (e.g., it takes only $O(1)$ table lookup to get the state output probability).

[3] Strictly speaking, due to the constraint on c_{jm}'s, P_j only lies in part of the subspace spanned by bases $B_1, \cdots, B_M$.

2.2.2. Connection with Other Models

SHDDHMM shares some resemblance with the semi-continuous or tied-mixture HMM (SCHMM). As a result, many techniques developed for SCHMM, such as adapting mixture weights for speaker adaptation [7], are also applicable to SHDDHMM. Despite the similarity, there are two key differences between the two models. Firstly, SHDDHMM is a discrete model, while SCHMM is a continuous model. Secondly, unlike SCHMM, the density pool and state-dependent weights of SHDDHMM only exist during the training stage. The final SHDDHMM for decoding only stores the state output discrete densities, and discards the pool and mixture weights.

The subspace concept of SHDDHMM also has some connection to the recently proposed subspace Gaussian mixture model (SGMM) [8], except that SGMM uses subspace technique to derive means of the state, while SHDDHMM uses it to generate the state output discrete table.

SHDDHMM is also related to subvector-quantized discrete mixture HMM (DMHMM) [6], which also uses subvector partition. However, in the final model, for each state, DMHMM still keeps a mixture of discrete tables, while SHDDHMM only stores a single discrete table. As a result, to get the state output probability during decoding, DMHMM needs to evaluate each discrete mixture, and then do the sumup, while SHDDHMM only takes one table lookup in $O(1)$ time, irrespective of the number of mixtures. Furthermore, the subspace technique to reduce parameters in SHDDHMM is not utilized in DMHMM.

2.3. Generation of SHDDHMM

Instead of flat-start training, a 1-stream SHDDHMM with M bases is first initialized from SCHMM, and is then reestimated. The basic steps of generating SHDDHMM are listed:

STEP 1: A 1-stream SCHMM with a global pool of M diagonal-covariance Gaussians is first trained.

STEP 2: Initialization of SHDDHMM: The mixture weights c_{jm}'s of SHDDHMM are copied from SCHMM. The SVQ discrete density tables B_{im}'s of SHDDHMM are initialized using SCHMM as:

$$B_{im}(l) = \frac{\sum_{t:SVQ_i(x_{it})=l} \gamma_t(m)}{\sum_{t=1}^{T} \gamma_t(m)} \quad (3)$$

where l is a SVQ codeword, $B_{im}(l)$ is the probability value from the entry indexed by l of SVQ discrete table B_{im}, $\gamma_t(m)$ is the posterior probability of observation x_t belonging to mixture m at time t given the SCHMM and the training observation sequence $x_1, \cdots, x_T$.

STEP 3: Retraining of SHDDHMM: The updating of SVQ discrete density tables in SHDDHMM uses exactly the same Eqn.(3), except that the posterior probability $\gamma_t(m)$ is based on SHDDHMM of the previous iteration, rather than SCHMM. SHDDHMM weights c_{jm}'s are also updated.

STEP 4: Generation of final SHDDHMM: After the reestimation is complete, the state output discrete densities P_j's are computed using the weights c_{jm}'s and SVQ discrete tables B_{im}'s by Eqn.(1) and (2). The final SHDDHMM only stores the state output discrete densities, and discards the weights and SVQ discrete tables.

Table 1. Baseline model performance on WSJ SI-84 (in WER)

System (Task)	CDHMM	DHMM	SCHMM
SI-84 (Nov'92 5K closed)	4.46	7.90	4.82

3. EXPERIMENTAL EVALUATION

The proposed SHDDHMM was evaluated on both RM and WSJ databases. For both databases, the conventional 39-dimensional MFCC vectors (with energy, delta, and delta-delta) were extracted.

The RM systems used the standard SI-109 training data, and were tested on Feb'89 and Feb'91 test set, with the standard word-pair grammar. The WSJ systems were trained using the WSJ0 SI-84 training data with 7,138 utterances, and evaluated on the Nov'92 5K closed vocabulary task, together with the standard 5K-word trigram language model. For each task, a development data set was set out to tune system parameters. All systems used cross-word triphones.

The HTK software was modified for SHDDHMM training and decoding. All experiments were run on a Linux machine, with a 3GHz CPU and 4GB RAM with results shown in WER. Due to the page limitation, we mainly discuss the experiments on WSJ task; For RM experiments, only the final WERs are reported.

3.1. Baseline Model Performance on WSJ SI-84

For SI-84, SHDDHMM was compared with 1-stream CDHMM, traditional 4-stream DHMM, and 4-stream SCHMM. The standard 4 streams were adopted in our 4-stream models: the 4 streams consist of the 12 static MFCCs, 12 ΔMFCCs, 12 $\Delta\Delta$MFCCs, and the 3 energies respectively. All models had 3,132 tied states. The CDHMM had 16 mixture components per state. The 4-stream DHMM had 512 codewords for each MFCC stream, and 256 codewords for the energy stream. Finally, the 4-stream SCHMM, which was used to initialize the SHDDHMM, had 1,024 tied mixtures per stream; only the top 128 Gaussians were used for its decoding (we found in the experiments that top 128 Gaussians for decoding in SCHMM is a good trade-off between accuracy and speed). For each model, we tried out best effort to figure out its optimal setting.

All the baseline model results are shown in 1. These results are comparable to others in the literature for the same task. As expected, compared with CDHMM, DHMM has a significant degradation in the recognition performance. SCHMM is close to CDHMM in WER, with only 0.36% absolute degradation.

3.2. Subvector-partition and Bit-allocation Schemes

4-stream SHDDHMMs of different subvector-partition and bit-allocation schemes were initialized from SCHMM. For fair comparison, all the SHDDHMMs had the same total number of bits for each stream, which were 16, 16, 16 and 15 respectively (i.e., the codebook size for stream 1 to 4 was 2^{16}, 2^{16}, 2^{16} and 2^{15} respectively, which resulted in a model size of roughly 1G bytes). We came up with these codebook size by considering both the memory and training data limit of our system. It can be seen that the codebook size of SHDDHMM is significantly larger than traditional DHMM, which was only 2^9, 2^9, 2^9 and 2^8 for each of the four streams.

Three subvector-partition and bit-allocation schemes used in the experiment are depicted in Table 2, with each of the last 3 columns representing a scheme. Each scheme describes a different subvector partition and its corresponding bit allocation gives the best WER of

Table 2. Different subvector-partition and bit-allocation schemes for 4-stream SHDDHMM

Streams	Scheme s1	Scheme s2	Scheme s3
Stream 1	$[1(2)]^4[1(1)]^8$	$3(6)3(4)[3(3)]^2$	$6(9)6(7)$
Stream 2	$[1(2)]^4[1(1)]^8$	$3(6)3(4)[3(3)]^2$	$6(9)6(7)$
Stream 3	$[1(2)]^4[1(1)]^8$	$3(6)3(4)[3(3)]^2$	$6(9)6(7)$
Stream 4	$[1(5)]^3$	$[1(5)]^3$	$[1(5)]^3$

Table 3. Different-Scheme SHDDHMM Performance on WSJ SI-84

Scheme	WER (Reestimation?)	
	no	yes
s1	6.82	6.11
s2	5.62	5.32
s3	5.53	**4.89**

that partition [4]. The notation of the scheme in Table 2 is explained as follows. The basic building block is $w(b)$, where w is the subvector dimension, and b is the number of bits allocated to the subvector. And $[w(b)]^C$ is equivalent to $\underbrace{w(b)\cdots w(b)}_{C\ in\ total}$. For example, in scheme $s2$, stream 1 (consisting of 12 static MFCCs) was split into four 3-dimensional subvectors, and 6 bits were allocated to the first subvector, 4 bits were allocated to the second subvector, and 3 bits were allocated to each of the last two subvectors.

In general, the subvector dimension increases from scheme $s1$ to $s3$. We did not try subvector dimension larger than the one in scheme $s3$, because the time of finding codewords increases significantly after that. (In the extreme case, SVQ becomes full-space VQ, and the time of finding codewords will be $O(N)$ where N is the size of the full-space VQ codebook.)

3.3. Effects of Schemes and Re-estimation on WSJ SI-84

Table 3 lists the SHDDHMM results with the three different subvector-partition and bit-allocation schemes. For reestimation, the column labelled with "no" gives results of the model that was initialized from an SCHMM without any reestimation, whereas the column labelled with "yes" gives the results of the final reestimated SHDDHMM. From the table, we observe that:

- From scheme $s1$ to $s3$, the WER keeps dropping. As expected, the performance is usually better when the subvector dimension is larger, as larger dimension results in smaller quantization error in full space.
- Initialization of SHDDHMM from SCHMM is effective (i.e., even without reestimation, the model is reasonably good), which provides a good starting point for reestimation.
- For each scheme, model reestimation always reduces the WER significantly.

3.4. Recognition Accuracy of Various Systems

Besides WSJ task, we also tested the models on RM task. The RM models consisted of 1,589 tied states; The CDHMM had 6 mixtures

[4] We did not exhaust all possible bit allocations. Instead, we use the heuristic of putting more bits to the lower-order MFCCs, making use of the fact that the lower-order MFCCs are more important for recognition task.

Table 4. Model performance on RM and WSJ (in WER)

System (Task)	CDHMM	SCHMM	SHDDHMM
RM (Feb'89)	2.85	2.77	2.66
RM (Feb'91)	2.94	2.94	2.98
WSJ SI84 (Nov'92 5k)	4.46	4.82	4.89

per state, and SCHMM had 512 tied-mixtures per stream. Similar procedures as in WSJ were used to build SHDDHMM. The final model performance on both RM and WSJ are given in Table 4.

As canbe seen, for each task, the WER of SHDDHMM is quite comparable to SCHMM. Since SCHMM could be viewed as a continuous version of SHDDHMM, it indicates that: the quantization error inherent in discrete model is largely alleviated in SHDDHMM, thus resulting in similar performance as continuous model.

On WSJ task, both SCHMM and SHDDHMM have some degradation compared with CDHMM, which is probably due to their incorrect stream independence assumption. However, all the degradations are not statistically significant according to the NIST matched pair sentence segment word error test with confidence level 0.95.

It is also worth pointing out that the degradation of SHDDHMM, compared with CDHMM, is less serious than some recently proposed discrete systems (e.g., [3]) on a similar task. In [3], the codebook size was not increased, and was comparable to the traditional one. Since a respectful amount of techniques were tried in [3] to reduce the quantization error with traditional codebook size, and yet the performance was still incomparable with that of CDHMM, it makes us believe that it is important to have a big codebook to make DHMM comparable with CDHMM.

3.5. Computation Time on WSJ SI-84

For the same number of states, we investigated the likelihood computation time for both CDHMM and SHDDHMM. The time ratio between the 2 models is found to be 40:1, which indicates 98% of GMM computation time could be cut off by SHDDHMM. This number is larger than most of other fast GMM computation techniques: in [9], chan implemented a set of fast GMM computation techniques, and found that 80% cut-off was an empirical upperbound.

The operating curves of different SI-84 models are compared in Fig. 2. The various operating points for a model were obtained by changing the beam width of the Viterbi beam search. The results show that SHDDHMM performs better than SCHMM and conventional DHMM. Compared with CDHMM, it cuts the decoding time by 25% relatively in most part of the operating curve. The 25% speedup is quite substantial, as in our CDHMM system, the Gaussian computation takes at most 30% [5] (i.e., the upperbound for speedup) of the total recognition time.

3.6. Bandwidth Saving

For CDHMM with 39-dim MFCC feature, each mfcc coefficient is represented by a 4-byte float, thus $4 \times 39 = 156$ bytes will be used to encode a frame; while for our 4-stream SHDDHMM, in each stream, the codeword is represented by a 2-byte unsigned short integer; As a result, only $2 \times 4 = 8$ bytes are used for a frame, which results in 20-fold saving of bandwidth in DSR systems. Even for systems which only transfer the 13-dim static MFCCs and construct the full 39-dim MFCCs out of it at the server side, the SHDDHMM feature could still result in 6.5 fold saving of bandwidth.

[5] In our experiment, HTK tool HDecode was used for decoding, with block caching mechanism to cache the state output likelihood, which saves gaussian computation time. As a result, we found that the Gaussian computation takes only less than 30% of total running time.

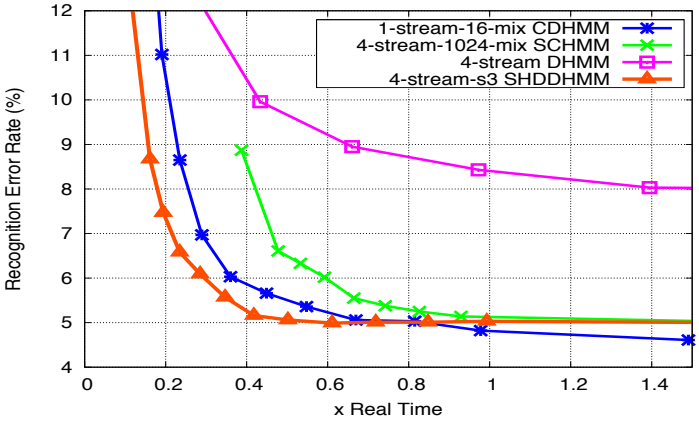

Fig. 2. Operating characteristics of various SI-84 models

4. CONCLUSION & FUTURE WORK

In this paper, we proposed to reduce the quantization error of DHMM with a very large codebook. Subspace modeling technique with constrained bases is applied to reduce the number of parameters in the model, which makes the re-estimation of such high-density model feasible. On both RM and WSJ 5K task, the proposed SHDDHMM gets similar recognition performance as that of CDHMM, with faster decoding speed and lower bandwidth requirement.

Further directions involve generalizing the model to larger tasks, where the size of SHDDHMM is expected to be even larger, and the stream independence assumption may hurt the recognition accuracy more.

5. REFERENCES

[1] M. J. F. Gales *et al.*, "State-based Gaussian selection in large vocabulary continuous speech recognition using HMMs," in *IEEE Trans. on SAP*, vol. 7, pp. 152–161, 1999.

[2] Jasha Droppo *et al.*, "Towards a non-parametric acoustic model: an acoustic decision tree for observation probability calculation," in *Proc. of Interspeech*, Sept, 2007.

[3] J. Ajmera and M. Akamine, "Decision tree acoustic models for ASR," in *Proc. of Interspeech*, Sept, 2009.

[4] Brian Mak *et al.*, "High-density discrete HMM with the use of scalar quantization indexing," in *Proc. of Eurospeech*, Sept 2005.

[5] Guoli Ye and Brian Mak *et al.*, "Subvector-quantized high-density discrete HMM and its re-estimation," in *Proc. of ISCSLP*, Nov 2010.

[6] V. Digalakis *et al.*, "Efficient speech recognition using subvector quantization and discrete-mixture HMMs," in *Computer Speech and Language*, vol. 14, pp. 33–46, 2000.

[7] X. Huang and K. F. Lee, "On speaker-indepenent, speaker-dependent, and speaker-adaptive speech recognition," in *IEEE Trans. on SAP*, vol. 1, pp. 150–157, 1993.

[8] Daniel Povey *et al.*, "Subspace Gaussian mixture models for speech recognition," in *Proc. of ICASSP*, Mar 2010.

[9] Chan, A. *et al.*, "Four-layer categorization scheme of fast GMM computation techniques in large vocabulary continuous speech recognition systems," in *Proc. of ICSLP*, 2004.