# Improved Performance of Aurora 4 using HTK and Unsupervised MLLR Adaptation

*Siu-Kei AU YEUNG, Man-Hung SIU*

Department of Electrical and Electronic Engineering
Hong Kong University of Science and Technology, Hong Kong
jeffay@ust.hk, eemsiu@ust.hk

## Abstract

The introduction of Aurora 4 tasks provides a standard database and methodology for comparing the effectiveness of different robust algorithms on LVCSR. One important issue on Aurora 4 tasks is the computation time involved in evaluating different test conditions. In this paper we show that by employing HTK as the recognition frontend and backend on Aurora 4 tasks with the use of cepstral mean subtraction, 14% relative improvement is achieved on the baseline clean train tasks at a 82.5% time reduction in training time and 40% time reduction on decoding. Furthermore, we found that optimizing the model complexity can increase the recognition performance (in both computation time and accuracy). Accuracy can be further improved with the use of unsupervised MLLR adaptation on one or multiple sentences. The adaptation results show that most of the gain from adaptation comes from adapting to the environment instead of to the speaker. With the use of adaptation,the error rate is reduced from the baseline result of 69.6% to 40%.

## 1. Introduction

For the automatic speech recognition technologies to be useful in real-world environments, the recognition system must maintain performance under varying noise and channel conditions. Different algorithms were proposed by researchers that showed various degrees of success. To allow an easy comparison between the results from different algorithms and researchers, a standard training and testing database with common evaluating criterion would be necessary.

The Aurora tasks [1] belong to these standard tasks and focus on noise-robust distributed speech recognition applications. The Aurora corpora and the associated standard evaluations have spurred tremendous research interest and progress [2]. The Aurora databases contain speech data with different noise types, noise levels and channels. Aurora 2 and 3 corpora both focus on small vocabulary continuous speech recognition tasks on digit speech. Aurora 2 contains only English speech data with artificially added noises while Aurora 3 contains speech from four different European languages recorded under real noisy environments.

As the complexity of ASR increases, research on robust speech recognition changes from small vocabulary tasks to large vocabulary tasks. The Aurora 4 database is part of the ETSI standardization process that evaluates the performance of robust techniques in LVCSR tasks in the presence of noise. Aurora 4 focuses on recognition using the standard SI-84 WSJ train-set and Nov'92 5000 words evaluation set. Six noisy environments and one clean environment are considered in the evaluation set under two different channel conditions. Two sampling frequencies (8kHz and 16kHz) are available for both training and testing data.

One main challenge in regard to robust LVCSR is the large number of tests to perform under different noise conditions. The computation of the test can grow quickly as the number of channel, noise type and SNR are enumerated. Thus, an efficient baseline system for training and decoding are vital. To address this issue, for example, the Aurora 4 scripts suggest the use of 4-mixture components in the HMM acoustic models to reduce experiment time.

The recognizer developed for the Aurora 4 is based on the system developed by ISIP [3] and ETSI WI007 frontend [4]. ISIP system provides a user-friendly interface to perform the whole training and evaluation process. Special features such as end-point detection, multi-CPU training and testing are included.

The hidden Markov model toolkit (HTK)[1] [5] is another widely used recognition system. In this paper, we describe our experience of using HTK to perform the Aurora 4 tasks and its effect on recognition accuracy and computational efficiency. Also, we describe the differences between the systems and how we handled parameter tuning and setting.

Furthermore, we investigated the effect of varying the model complexity as well as the use of MLLR adaptation for noise robustness. The use of MLLR for noise robustness has been discussed by other researchers. In our experiments, we focused on un-supervised adaptation and endeavoured to decouple the effect of adapting to the speaker and adapting to the environment.

The organization of this paper is as follows. In Section 2, an analysis of the baseline system on Aurora-4 using HTK is presented. In Section 3 we present how model complexity relates to robustness in Aurora 4. The study of noise adaptation on Aurora 4 using MLLR is discussed in Section 4 and the paper is concluded in Section 5.

## 2. Aurora 4 Baselines

### 2.1. Aurora 4 Baseline Using the ISIP System

The standard Aurora 4 task evaluation is based on the standardized ETSI mel-cepstrum front-end(ES 201 108) and Mississippi State University (ISIP) recognizer as the backend. Four mixtures cross-word tripohone models was trained in both clean and multi conditions. 14 testsets with different noise and channel conditions are evaluated. In our experiment, we focused on the tasks of 16kHz speech without data compression. The 14 testsets are grouped into the following four

[1] We used version3.2

families with the number inside the brackets representing the set number defined in Aurora 4 databases.

1. Testset A : clean data in the same channel as in training (set 1)

2. Testset B : noisy data in the same channel as in training (sets 2 to 7)

3. Testset C : clean data in a different channel as in training (set 8)

4. Testset D : noisy data in a different channel as in training (sets 9 to 14)

The results of our reproduction of the Aurora 4 standard baseline of clean and multi condition training data are tabulated in Table 1 and Table 2 respectively. They generally matched the result presented in [3].

Table 1: *Aurora 4 standard baseline clean train (frontend: ETSI, backend: ISIP)* .

| Testset | A | B | C | D | Avg. |
|---|---|---|---|---|---|
| WER | 14.5% | 69.3% | 60.6% | 81.43% | 69.6% |

Table 2: *Aurora 4 standard baseline multicondition train (frontend: ETSI, backend: ISIP)* .

| Testset | A | B | C | D | Avg. |
|---|---|---|---|---|---|
| WER | 23.5% | 31.3% | 47.4% | 48.8% | 39.3% |

### 2.2. Experimental design of Aurora4 tasks using HTK

In our other work [6] and in the Aurora 2 and 3 experiments, HTK was used as the standard system for training and decoding. Because of this, it may be also of interest to the community to see the performance of using HTK on the Aurora 4 task.

Our experimental setup on Aurora4 used HTK for both frontend and backend[2], that is, the ETSI frontend was replaced by the HTK frontend using similar frontend parameters. The feature extraction process was generally the same between ETSI and HTK except that HTK frontend performs liftering. HTK also allows for cepstral mean subtraction and the results are described in a later section.

For the recognition backend process involving model training and decoding, HTK replaced the ISIP recognition system. We made the training setting used in both systems as close as possible to those mentioned in [3]. In state clustering, 3215 tied states were used in HTK compared to 3202 states reported in [3]. However, to speed up training in HTK, pruning was applied. In decoding, the settings of the two systems (HTK and ISIP) were made as close as possible. The main difference between the two system was the pruning. The ISIP ASR support three levels of pruning (state, phone and word) while HTK only supports state pruning. The state pruning width was set to 300 which is larger in numerical value than the setting in ISIP's. In addition, whenever pruning error occurs such that the recognition cannot reach the end of the sentence for a valid hypothesis, HTK does not generate any hypothesis thus results in a whole sentence being deleted while the ISIP system generates a partial decoding result. In Table 3 the key features of the HTK baseline system on Aurora 4 compared with the ISIP system are summarized.

[2] scripts available at http://ihome.ust.hk/~jeffay/Aurora4_htk.tgz

Table 3: *Summary of HTK baseline system on Aurora 4.*

| | |
|---|---|
| Running Machine | P4-2.4GHz |
| Data Storage | a separate file sever |
| Frontend | CMS is performed and dynamic parameters are generated |
| Backend | training and decoding using HTK |
| training process | 4 mixture xwrd-triphone with pruning=250 |
| decoding process | state pruning=250 and blank pruned sentences during evaluation |

### 2.3. Performance Comparison between HTK and ISIP systems

We started the HTK experiments by using a frontend without CMS and a prune width of 300. In Table 4 the clean training result of Aurora 4 task using the HTK system is summarized. Compared with the result using ISIP system (Table 1), an overall of 7.3% improvement is obtained. This is somewhat surprising since both system were set up in more or less the same way. The only uncertainty may be the prune width used.

Table 4: *HTK baseline clean train(frontend: HTK, backend: HTK) no CMS, 4mixture, pruning=300* .

| Testset | A | B | C | D | Avg. |
|---|---|---|---|---|---|
| WER | 13.1% | 62% | 60.1% | 76.7% | 64.7% |
| Relative imp. (ISIP) | 9.7% | 10.5% | 1.9% | 5.7% | 7.3% |

We improved our baseline of Aurora 4 by applying the Cepstral Mean Subtraction(CMS) on the training data and testing data. In order to speed up the process, the prune width on decoding was decreased from 300 to 250. The results for clean training and multi-condition training are tabulated in Table 5 and Table 6. For the clean training set, another 7.5% improvement is obtained. The improvement, more than 30% is particularly significant in Testset C. The result suggests that CMS is effective in reducing part of the channel effect in the Aurora 4 task. However, the accuracy of the test-set C is still significantly worse than test-set A which has the matched channel.

A similar overall improvement was obtained in the multi-condition experiment when using HTK and CMS.

As clean training has a more significant mis-match between test and training and because this can demonstrate the usefulness of the algorithms, we focus on the clean train experiment in the rest of this paper using the result in Table 5 as a baseline.

In addition to comparing the recognition accuracy, we also compare the processing time between HTK and the Aurora suggest ISIP recognizer system. The timing results are summarized in Table 7. Because we used a more powerful computer than reported in [3], we needed only 40 hours to train a four mixtures cross-word triphone model using the ISIP trainer. However, using the same type of machine, training in HTK took only seven hours, about one-sixth the time needed for the ISIP system. This may have been due to the effect of using pruning in the HTK training. As the performance of HTK is better than the baseline, the pruning in training does not appear to affect performance.

In the decoding, similar to the training, an increased computing power reduced the decoding time for the Nov 92 clean

data (330 test data in testset 1)from 50 CPU hour [3] to about ten hours when using the ISIP recognition system. The decoding was further speeded up to six hours using the HTK at a prune with of 250.

Table 5: *HTK baseline clean train(frontend: HTK, backend: HTK) CMS, 4mixture, pruning=250 .*

| Testset | A | B | C | D | Avg. |
|---|---|---|---|---|---|
| WER | 11.2% | 56.8% | 42.2% | 73.6% | 59.7% |
| Relative imp. (ISIP) | 22.8% | 18% | 30.4% | 9.6% | 14.5% |

Table 6: *HTK baseline multicondition train(frontend: HTK, backend: HTK) CMS, 4mixture, pruning=250 .*

| Testset | A | B | C | D | Avg. |
|---|---|---|---|---|---|
| WER | 18.1% | 26% | 30.4% | 40.6% | 32% |
| Relative imp. (ISIP) | 17.1% | 16.9% | 35.9% | 16.8% | 19.2% |

Table 7: *Processing time on Aurora 4 tasks.*

| | ISIP | HTK |
|---|---|---|
| Training (clean train) | 40 hours | 7 hours |
| Testing (standard Nov'92 330) | 10 hours | 6 hours |

## 3. The issue of model complexity

As mentioned in [3], the original Aurora 4 tasks were designed to use 16-mixture, cross-word triphone models. Because high computational time have been the result, the model complexity is finally reduced to four mixtures. The WER on clean data evaluation (Testset A) is increased by 20% after the mixture reduction [3]. However, no further analysis is performed on the effect of the noisy data. It is well-known that, although the likelihood of the training data increases monotonically as the model complexity increases, the likelihood of the testing data may actually decrease due to over-training. This is especially true in the mis-matched conditions. A highly optimized system may perform worse. In our experiment, we investigated how the robustness changes with model complexity.

We studied an extreme case by reducing the model from four mixtures to one mixture. All other settings remained the same as in the four mixture experiment. Table 8 shows the performance of using the one mixture models. Compared with the four mixtures model, only the testset A, the clean data with the same channel, degrades. The other three testsets perform as well as, or better than, the four mixtures model. In addition, using the one mixture models reduces the decoding time by 50%. The result is consistent with our expectation that models with less mixtures have higher variances and thus, are more robust to mis-match at the price of less accurate modeling. The less accurate modeling causes the clean test (testset A) result to degrade. However, in the other testset, it seems that the robustness is sufficient to compensate for the lost in accuracy. This suggests that one way to deal with noisy test data is to vary the number of mixtures during the test with less mixtures in higher mis-match, and more mixtures when clean.

Table 8: *HTK clean train with 1 mixture model(frontend: HTK, backend: HTK) CMS , pruning=250*

| Testset | A | B | C | D | Avg. |
|---|---|---|---|---|---|
| WER | 16.5% | 57.5% | 43.2% | 72.7% | 60.1% |
| Relative imp. (ISIP) | -13.8% | 17% | 28.7% | 9.6% | 13.9% |
| Relative imp. (HTK) | -47.32% | -1.2% | -2.3% | 1.6% | -0.6% |

## 4. Noise adaptation in Aurora 4 task

Instead of changing the model complexity, one can transform the models to match the test conditions. In this paper, we report the result of using the Maximum Likelihood Linear Regression (MLLR) [7], which is widely used in speaker adaptation tasks, in the Aurora 4 tasks. MLLR has been applied in environment adaptation [8].In speaker adaptation, non-speech models, such as silence and short pause, are typically not adapted because they are not speaker dependent. However, in noise and channel adaptation, accurate non-speech models can improve speech alignment during the decoding, and reduces insertion error significantly.

### 4.1. Unsupervised MLLR with 1 utterance as adaptation data

For one utterance unsupervised MLLR, the testing utterance was first decoded with the clean model and the decoded phone transcription obtained was then used to adapt the clean model based by the MLLR technique. Only the Gaussian means were adapted with a 32-node regression tree. The first split of the regression class tree separates the speech and non-speech models. That is, the non-speech models (silence, short pause) and speech sounds were adapted separately. The testing utterance was decoded again using the adapted model.

Table 9 tabulates the result in an unsupervised MLLR with one utterance of adaption data. A significant improvement was obtained. In particular, testsets B & D (Noisy data in two channels) improved by more than 12%. Although the testing utterances needed to be decoded twice, the second decoding using the adapted models was much faster than the first one because of the model improvement resulting in the average processing time being increased only by about 20%.

When we compared the result on Table 5, the testset A (clean test data) degraded a little bit (from 11.2% to 11.3%). The fact that the clean data do not contain any noise or mismatch channel, it suggests that speaker adaptation with only one utterance does not improve the performance much. However, more significant improvement was obtained in Testset C, clean data with different channel suggesting that the gain comes from channel adaptation. To understand the relationship between amount of data, speaker and environmental effect in adaptation, we further tested MLLR adaptation as reported in the next section.

### 4.2. Unsupervised MLLR using 3 utterance of adaptation data

To see how significant more data can help in adapting to the environment, the number of adaptation utterances was increased to three. After the three utterances were decoded with the clean model, adaptation was performed in two settings. In the first set, three utterances from the same speaker were used for adap-

Table 9: *MLLR adaptation (1 utt)* .

| Testset | A | B | C | D | Avg. |
|---|---|---|---|---|---|
| WER | 11.3% | 48.7% | 35.4% | 64.4% | 51.8 |
| Relative imp. (ISIP) | 22% | 29.7% | 41.6% | 20.9% | 25.9% |
| Relative imp. (HTK) | -0.1% | 14.3% | 16.1% | 12.5% | 13.2% |

tation. the aim was to increase the speaker characteristics in the adaptation data and show how much gain the combined speaker and environmental adaptation can attained via MLLR. In the second set, three utterances from three different speakers selected randomly in the testset were used as adaptation data. The aim was to minimize the speaker effect but allowing more data for environmental adaptation.

Table 10 and Table 11 show the results of the unsupervised MLLR using three utterance of adaptation data from the same speaker and different speakers respectively. It is clear to see that increasing the adaptation data gives a much better result. In testset A, a 5% improvement is obtained when the adaptation data comes from the same speaker while no improvement is obtained when using three different speakers. Thus, as expected, speaker effect is minimized by using three different speakers and the gains for testset B,C and D can be argued to have come solely from environmental adaptation. Comparing the results in Table 10 with Table 11, additional gain of 3-8% were resulted for testsets B and C because of speaker adaptation. This shows that a significant larger portion of the gain comes from adapting to the noise and channel environment. However, for testset C, The MLLR using three utterances from different speakers performed poorly and is not consistent with the other results. We are still investigating the underlying reasons.

Table 10: *MLLR adaptation (3 utts from same speaker)* .

| Testset | A | B | C | D | Avg. |
|---|---|---|---|---|---|
| WER | 10.7% | 35.5% | 24.4% | 52% | 40% |
| Relative imp. (ISIP) | 26.2% | 48.8% | 59.7% | 36.1% | 42.8% |
| Relative imp. (HTK) | 4.5% | 37.5% | 42.2% | 29.3% | 33% |

Table 11: *MLLR adaptation (3 utts from different speakers)* .

| Testset | A | B | C | D | Avg. |
|---|---|---|---|---|---|
| WER | 11.2% | 38.1% | 39% | 60% | 45.6% |
| Relative imp. (ISIP) | 22.8% | 45% | 35.6% | 26.3% | 34.8% |
| Relative imp. (HTK) | 0% | 32.9% | 7.6% | 18.5% | 23.6% |

## 5. Conclusion

In this paper, we report our experience of performing Aurora 4 tasks using HTK. We have shown that using HTK significantly improve the the accuracy of Aurora 4 baseline evaluation and can be performed with shorter processing time. In addition, the use of sentence-level CMS can significantly improve the robustness of the model. We further investigate the relationship between model complexity and robustness, showing that a simple single mixture model which requires significantly less processing time, can perform as well as the 4-mixture models under noisy conditions. We believe that by optimizing the model complexity, a good balance between the performance and processing time can be achieved. In addition, we show that noise adaptation can be performed using MLLR technique resulting in significant improvement. By performing adaptation using data from different speakers to remove the contribution of speaker adaptation, we showed that the gain from MLLR comes mostly from noise and channel adaptation while speaker adaptation contributed less.

As the processing time is reduced, it may be possible to revert back to using the 16-mixture model as originally suggested. Using 16-mixture models may also increase the flexibility of optimize the model complexity. Further study will be focused on how to obtain an optimal model complexity and how to improve the noise adaptation in terms of accuracy and processing time.

## 6. Acknowledgement

This work is partially supported by Hong Kong RGC, under the Central allocation Grant number HKUST CA02/03.EG05.

## 7. References

[1] H. G. Hirsch and D. Pearce, “The AURORA experimental framework for the performance evaluations of speech recognition systems under noisy conditions,” in ISCA ITRW ASR2000 “Automatic Speech Recognition: Challenges for the Next Millennium”, Paris, France, September 2000.

[2] Jasha Droppo, Li Deng and Alex Aero, “Evaluation of SPLICE on the Aurora 2 and 3 tasks,” International Conference on Spoken Language Processing, September 2002, pp.29-32

[3] N. Parihar and J. Picone, “DSR Front End LVCSR Evaluation,” AU/384/02, Aurora Working Group, Dec. 2002

[4] “ETSI ES 201 108 v1.1.3 Speech Processing, Transmission and Quality Aspects(STQ); Distributed speech recognition; Front-end feature extraction algorithm; Compression algorithms,” ETSI, September 2003

[5] Steve Young, Gunnar Evermann, Dan Kershaw, Gareth Moore, Jukia Odell, Dava Ollason, Dan Povey, Valtcho Valtchev, Phil Woodland, “The HTK Book (for HTK Version 3.2),” Microsoft Corporation, Cambridge University Engineering Department,December 2002

[6] Yiu-Pong Lai and Manhung Siu, “Maximum Likelihood Normalization for Robust Speech Recognition,” Eurospeech 2003, Geneva, Switzerland, Sep, 2003

[7] C. J. Leggetter, P. C. Woodland, “Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models,” Computer Speech and Language 9, 1995 pp 171-186

[8] Prabhu Raghavan, ”Speaker and Environment Adaptation in Continuous Speech Recognition,” CAIP Technicak Report No. TR-227, 1998.