Phone Based Continuous Speech Recognition with GMM-HMMs and DNN-HMMs

Chris Tegho

February 1, 2017

1 Introduction

This project is concerned with phone-based continuous speech recognition using Hidden Markov Models (HMMs). The use of Gaussian mixture models (GMM) to form the state output distributions and the use of Deep Neural Networks (DNNs) in a hybrid configuration are investigated. The aim is to develop a speaker independent phone recognition system for the TIMIT Acoustic-Phonetic speech database. HTK version 3.5 [1] and scripts specifically supplied for the practical were used for the experiments. This report is organized as follow: setup and results are reported first for GMM-HMMs with monophones and triphones, then for DNN-HMMs. Biphones and the bigram language model are then investigated and observations and results are discussed. The last section summarizes results from all experiments and provides further discussion on the comparison between all models.

2 Monophones With Different Front End Parameterisations

2.1 Setup

2.1.1 Feature Types

Two main feature types were supplied:

- 24 channel filter bank (FBANK)
- 12 Mel-Frequency Cepstral Coefficients with normalized log energy (MFCCs)

Filterbank analysis provides a non-linear frequency resolution. Mel-Frequency Cepstral Coefficients (MFCCs) are calculated from the log filterbank amplitudes. Original feature vectors all contain sentence-based mean removal (Z) and delta coefficients (1st and 2nd differentials) were added to each feature type. The MFCC $E\_D\_A\_Z$ is a 39-D feature vector.

2.1.2 Initialization and Training

There are 39 monophones and 1 silence model. Each phone is modeled with a HMM that has three output states with a left-to-right topology with self-loops and no transitions which skip over states. Each HMM is **initialized** using the HTK tool HINIT using phonetically aligned utterances in TIMIT. Viterbi-training is then used iteratively until the parameters converge. An additional Baum-Welch training cycle refines those parameters [2].

This initialization requires labeled training data [1]. A **flatstart** is an alternative initialization where all models are initialized with the global mean and variance of the training data. Utterances are then uniformly segmented and their parameters re-estimated in several rounds of Baum-Welch training.

To increase the number of Gaussians per state, each Gaussian is split. The new models are trained with 4 rounds of Baum-Welch and the splitting and training process is repeated. I experiment with 1,2, 4 ... up to 20 Gaussians per non-silence state. The silence models use double the number of output distributions.

2.2 Experimental Findings

2.2.1 Filter Bank vs MFCCs

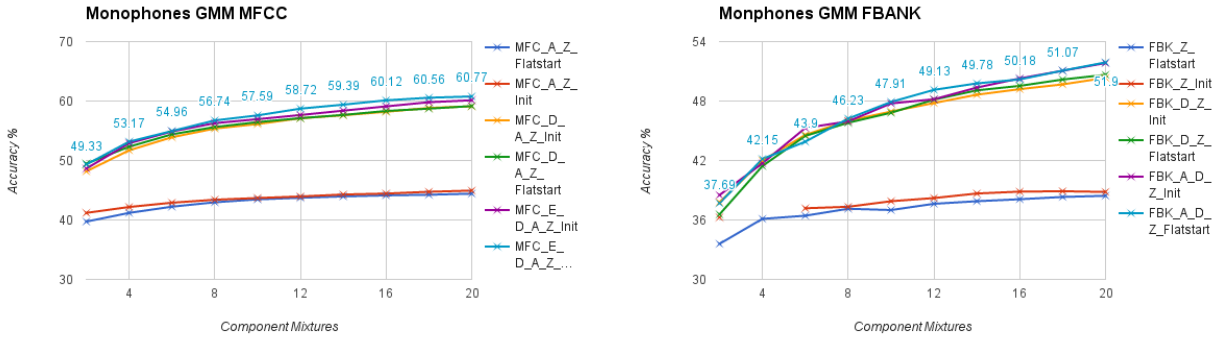
MFCC provide higher performance over filter bank features. This is because cepstral features are more compact, discriminable, and nearly decorrelated such that they allow the diagonal covariance to be used by the hidden Markov models (HMMs) effectively. Filterbank amplitudes are highly correlated and it is necessary to use a cepstral transformation for a HMM based recogniser with diagonal covariances [3]. Delta coefficients provide additional improvement since they capture more dependency between each of the feature vectors.

2.2.2 Number of Mixtures

The single Gaussian output distribution model assumes that the observed feature vectors are symmetric and unimodal. Speaker, accent and gender differences tend to create multiple modes in the data which are not modeled well with a single Gaussian state-output distribution . Instead, a mixture of Gaussians is used which is a highly flexible distribution able to model asymmetric and multi-modal distributed data [4]. Increasing the number of mixtures improve phone recognition results considerably, for all feature types. While the model is expected to fit more to the training data as the number of mixture components is increased, overfitting did not occur in the experiments. This is mainly because the mixtures are augmented in stages, two at a time. Recognition accuracy increases as we increased the number of mixture components for both the training and testing set. The accuracy saturates faster on the test set.

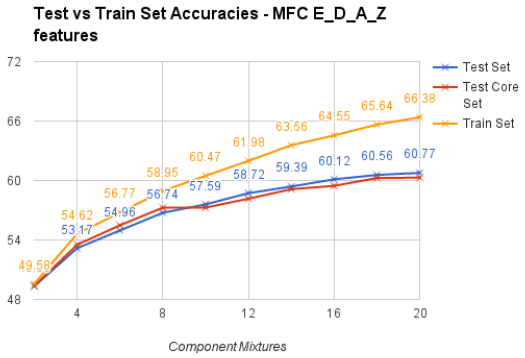
2.2.3 Initialization

Performance was slightly higher for both MFCC and FBANK input features when building the monophones with a flat start. A flat start is less dependent to the existing model and allows better generalization when decoding on the test set.



(a) (b)

Figure 1: Performance accuracy for monophones with GMM-HMMs, with (a) MFCC and (b) FBANK input features, with different acoustic context. Results are for the test set.

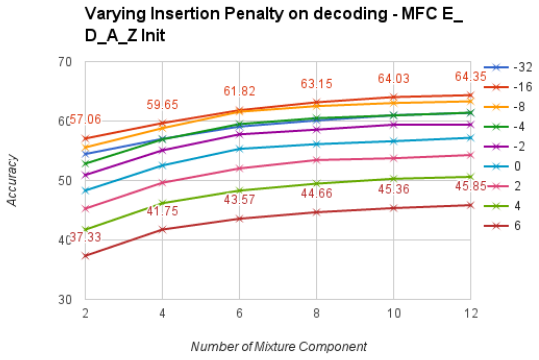


(a)

Figure 2: Performance accuracy for monophones with GMM-HMMs, with MFCC input features, with different acoustic context. Results are for the train set.

2.2.4 Insertion Penalty

To control the relative levels of insertion and deletion errors, a word insertion penalty is added. Overall, a negative value of the penalty results in less word transitions and accuracy increased when the insertion penalty decreased to -16. However, further decreasing the insertion penalty to -32 gave worse results. Positive insertion penalties decreased accuracy, mainly because they result in many short words at the output.



(a)

Figure 3: Performance accuracy for monophones with GMM-HMMs, **varying the insertion penalty** for MFCC EDAZ input features with initialization.

3 Triphone Decision Tree State Tying with GMM-HMMs

3.1 Setup

3.1.1 Context Dependent Model

One issue with using context independent phones or monophones is the variability in acoustic realisation due to co-articulation. Co-articulation can be modeled by using context dependent phone models. A unique phone model for every possible pair of left and right neighbors is used instead, resulting in triphones. For N base phones, there are N^3 potential triphones. To build the triphones, monophone transcriptions are converted to triphone transcriptions and a set of triphone models are created by cloning monophones with well-performing front-end parameterization and re-estimating [1].

3.1.2 State Tying

Many triphones may not have been observed in the training data. To avoid data sparsity problems and ensure all state distributions can be robustly estimated, similar acoustic states are tied using a decision tree which was provided for this project and which includes yes/no questions attached to nodes. Depending on the answer to the questions, pools of states are split until states reach leaf nodes. Those states are then tied.

Single Gaussians distributions are used for an initial untied model set. Tying is then performed, increasing the amount of data available per state and higher order mixture Gaussian distributions are estimated. In HTK, compared to data-driven methods, tree based clustering provides a mapping for unseen triphones and generally leads to better results when expert phonetic knowledge is supplied [5].

Crossword triphones were used and the resulting tied-state triphones were re-estimated using 4 rounds of Baum-Welch training. Clustering and tying the models at the state level (compared to tying at the level of the model, the variance matrix or the output PDFs) is simpler and "allows a larger set of physical models to be robustly estimated" [4].

3.1.3 Tying Thresholds

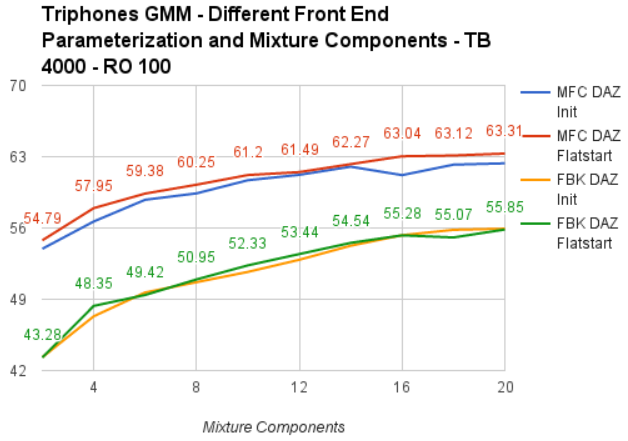
RO. To prevent creating clusters with very little associated training data, a clustering threshold RO is used and controls the number of outliers or leaf nodes. The final number of clustered states depends on the RO

value chosen. "Any split which would result in a total occupation count falling below the value specified is prohibited" [1]. **TB**. The question is then found which gives the best split of the root node. This process is repeated until the increase in log likelihood falls below the threshold specified in the TB command.

3.2 Observations and Findings

3.2.1 Input Features

As for monophones, MFCC input features with a flatstart for training outperformed FBANK input features.

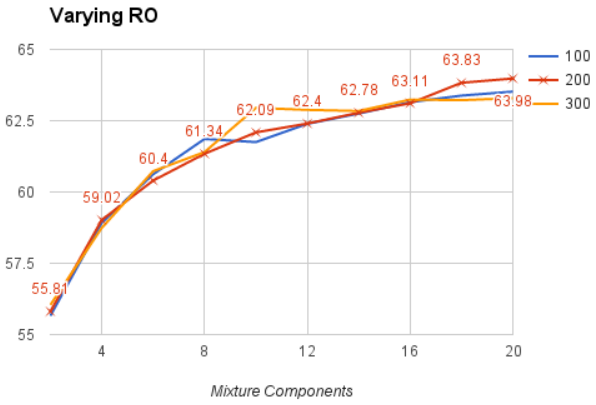


(a)

Figure 4: Performance accuracy for triphones with GMM-HMMs, with different input features.

3.2.2 RO

For a fixed value of TB, changing RO did not have a significant effect on performance.

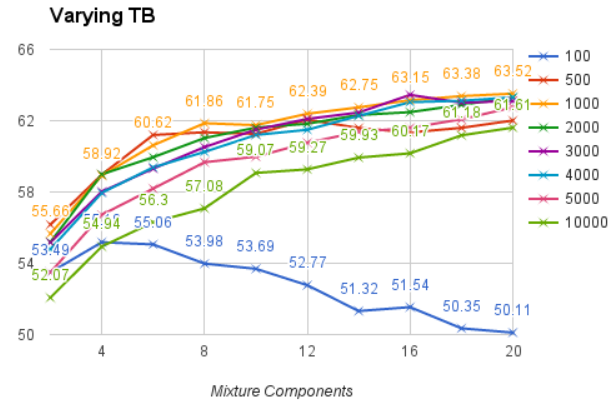


(a)

Figure 5: Performance accuracy for triphones with GMM-HMMs, with MFCC EDAZ input features. The TB value was kept fixed to 1000 and the **RO** value changed.

3.2.3 TB

For a fixed value of RO, overall, decreasing the value of TB from 10000 down to 100 led to better results for all models considered in the experiment. For values of TB below 1000, performance decreased. The decrease was significant when the TB value was brought down to 100 and when the number of Gaussians per state increased. As the number of Gaussians mixture components per state increases, the number of parameters to estimate increases to a level where the training data is not sufficient to obtain a robust estimation of the parameters.



(a)

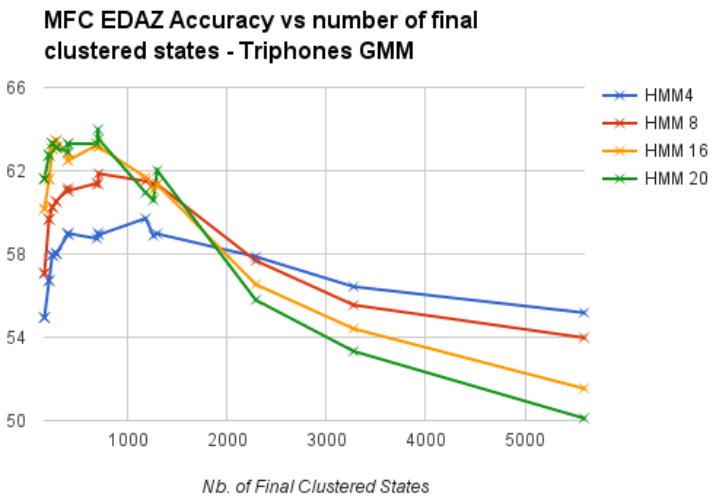
Figure 6: Performance accuracy for triphones with GMM-HMMs, with MFCC EDAZ input features. The RO value was kept fixed to 100 and the **TB** value changed.

3.2.4 Number of Clustered States

As the values given in the RO and TB commands decrease, the number of possible cross -word triphones and the number of states output in the clustered system increase. Table 1 shows the total number of final states for different values of TB and RO. As the number of cross-word triphones increases, a larger number of these triphones will have few, if any, occurrences in the training data. As shown in Figure 6 and 7, accuracy was highest with about 11178 (RO 300 TB 500) clustered sates when estimating parameters with up to 10 Gaussians distributions, and about 705 (RO 100 TB 1000) clustered states for an increased number of Gaussian mixtures.

Table 1: Number of final states for different values of RO and TB commands which control the threshold for triphone decision tree state-tying with GMM-HMMs.

RO	TB	Final States (from 60201)
100	100	5594
200	100	3273
300	100	2288
100	500	1295
200	500	1257
300	500	1178
100	1000	705
200	1000	698
300	1000	689
100	2000	396
200	2000	396
300	2000	395
100	3000	278
200	3000	278
300	3000	278
100	4000	239
200	4000	239
300	4000	239
100	5000	206
200	5000	206
300	5000	206
100	10000	159
200	10000	159
300	10000	159



(a)

Figure 7: Performance accuracy with GMM-HMMs models for triphones. Results are shown for **a varying final number of clustered states** for the MFCC EDAZ input features.

TB and RO. Overall, smaller values of TB and RO yielded better results, although smaller thresholds lead to a larger final number of clustered states. In general, a smaller number of clustered states is desired to avoid issues related to unseen triphones. For a value of TB larger than 500, larger number of final clustered states (up to a total of 705 tied states) led to better results. The amount of data in the training set was sufficient enough to allow a robust estimation of the model parameters. Increasing the number of clustered states past 1178 decreased performance, especially for large number of mixture components. At that level, the training data was not sufficient to estimate the increased number of model parameters.

3.2.5 Component Mixtures

Overall, a higher performance was achieved with a larger number of mixture components. However, the highest accuracy was obtained with MFCC EDAZ features with 12 mixture components, a TB value of 1000 and a RO value of 100. While increased number of Gaussians achieved very similar results, the recognition accuracy saturated for the given amount of training data. To achieve (considerably) higher accuracy with higher number of Gaussians, more training data is necessary.

Both increasing the number of Gaussians and increasing the number of tied states needs more data. In this situation, the amount of training data was sufficient enough to allow robust model parameters estimation for large number of tied states (up to 1178) and large number of Gaussians (up to 12 Gaussians).

4 DNN-HMMs

This section investigates the use of DNN-HMMs for TIMIT and examines various configurations of ANNs, features and acoustic context. The number of hidden layers, of nodes per hidden layer were varied and the use of CI and CD output targets were compared. All DNN-HMM training uses the frame-based cross-entropy objective function.

4.1 Setup

The neural network is trained using a state alignment that provides a label for each frame in the training set. The alignments used were generated from the GMM-HMM with acoustic context that achieved highest accuracy: The MFCC and FBANK input features with 1st and 2nd differentials (EDAZ) (although alignments with simpler acoustic contexts were used for monophones to study the effect of the the context width). The first experiment used the context-independent phones for the output layer. After a suitable configuration was determined, experiments with context dependent output layer were run.

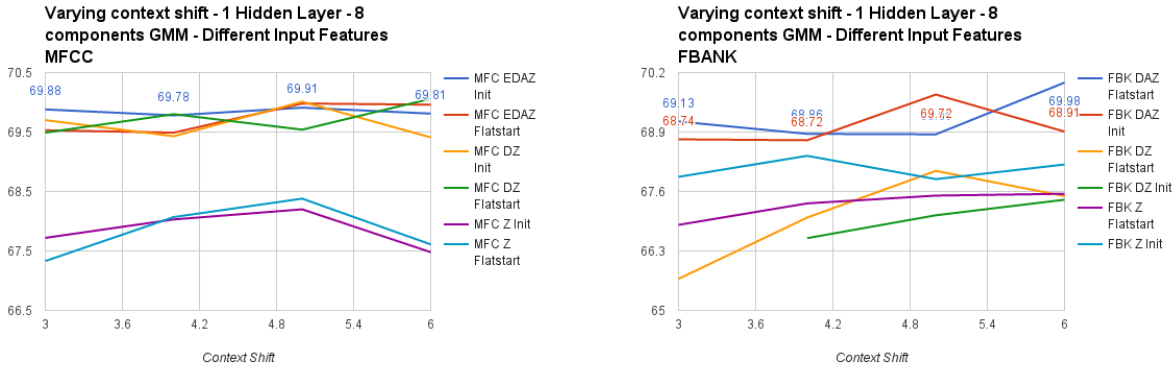
4.2 Monophone Target Units

4.2.1 Single Hidden Layer Experiments

The **context width** and the **feature type** were investigated jointly, for a DNN with a single hidden layer with 500 nodes per layer. Increasing the context width did not result in significant improvements when alignments from GMM-HMMs with input features with 2nd and 1st differentials were used. However, when the context width was increased to 5, for alignments obtained with MFCC input features with 1st differentials, performance slightly increased and became similar to the one for features with 2nd differentials. Increasing this value further to 6 slightly decreased performance in that case. Changes in performances were more arbitrary for alignments from the FBANK input features, but generally, increased performance was observed with increased context shift. Highest performance was achieved with alignments obtained with the FBANK DAZ input features with a context width of 6 frames.

Input Features: Similar performances were obtained for alignments fom MFCC and FBANK input features. Alignments with features with 2nd order and 1st order differentials outperformed ones with features with no additional acoustic context.

Context Width: Overall, context shift has a bigger effect on alignments with input features with no delta parameters. Increasing the acoustic frames window allowed better performances. This is because DNNs do not make much prior assumptions about the input feature space. Adding delta parameters to the input features for the alignments or augmenting the context width allow to take into account temporal and frequency structural localities [6]. In general, as noted by [7], smaller context width miss important discriminative information, while larger windows might include irrelevant information far from the center of the window.

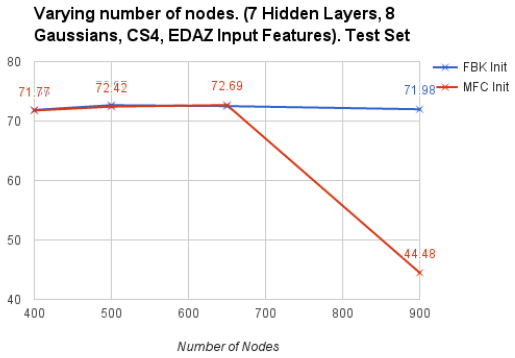


(a) (b)

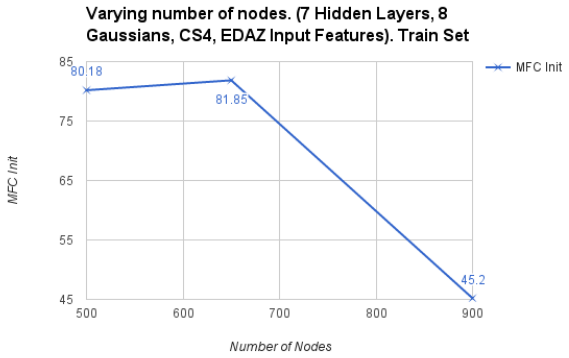
Figure 8: Performance accuracy with DNN-HMMs models for monophones. 8 Gaussian components per state were used and results are shown for **varying context width** for alignments obtained with **different acoustic contexts** for (a) the MFCC and (b) the FBANK input features. Results reported for the test set.

4.2.2 Varying Network Width (Nodes)

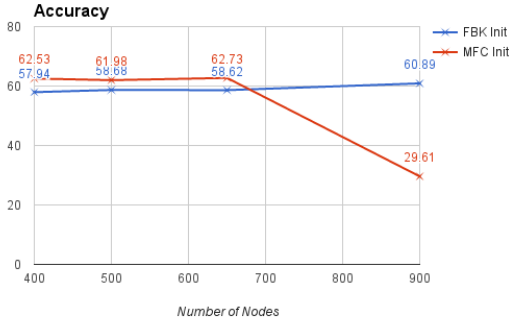
To evaluate the effect of increasing the number of nodes, a 7 hidden layers model with CI targets, and alignment from monophones with FBANK and MFCC input features with 1st and 2nd differentials were used. Incrementing the number of nodes only slightly improved the phone recognition accuracy. For MFCC input features alignments, the recognition rate dropped significantly when the number of nodes was increased to 900. Such drop was observed on the training set as well, and the CV frame classification performances. While it is possible to believe such drop is due to overfitting on the training data, the drop was present in the training set as well. This suggests that as networks become very wide, the total number of parameters in the system increases, and the estimation of the parameters becomes more difficult [8].



(a)



(b)

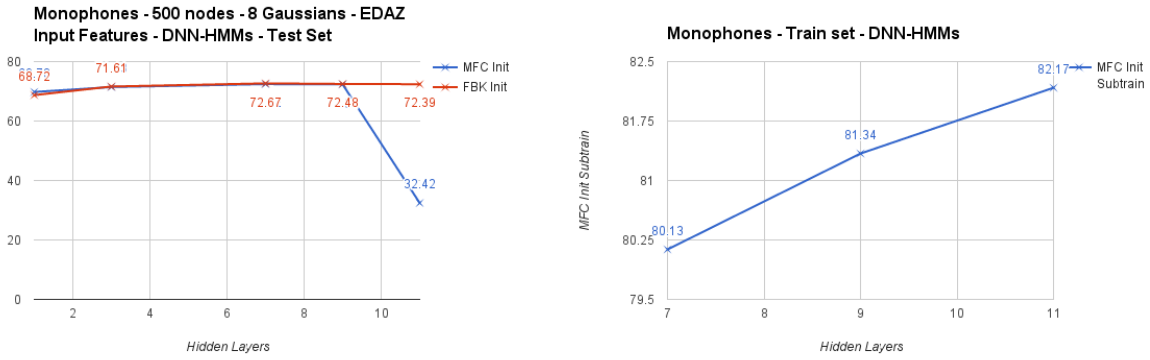


(c)

Figure 9: Performance accuracy with DNN-HMMs models for monophones. Number of hidden layers: 7, context shift: 4, input features with 1st and 2nd differentials and 8 Gaussians components per state were used for the alignments. Results are shown with **varying number of nodes** for (a) the test set and (b) the training set and (c) the CV frame classification.

4.2.3 Adding Multiple Layers

The recognition accuracy increased when we increased the number of hidden layers to 9. The improvements were not significant however. The accuracy dropped significantly when the number of layers increased to 11, for the MFCC input features. Such drop was not observed for the training set suggesting that the model overfits on the training set when the number of layers increases. This drop could also be explained by the increase in number of parameters to estimate and the difficulty to do such estimation. As the number of parameters to estimate increases, with a fixed number of epochs and with Stochastic Gradient Descent, parameters are more likely to be stuck at poor local optima. It might be possible to find better parameter solutions and perform higher accuracy with more hidden layers by performing more parameter updates (SGD might get out of the poor local optima after more iterations). Full batch gradient descent instead of SGD or an activation function with ReLU instead of sigmoid can be used as well. However such solutions have higher computation costs and can be difficult to apply for large scale problems.



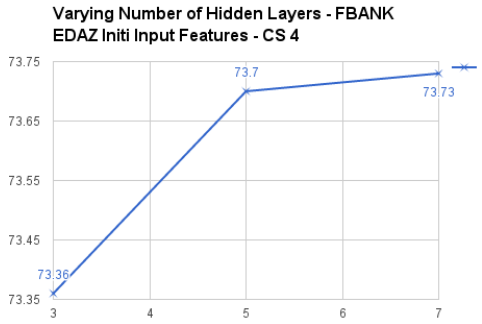
(a) (b)

Figure 10: Performance accuracy with DNN-HMMs models with monophone target units. Number of nodes: 650, context shift: 4, input features with 1st and 2nd differentials and 8 Gaussian components per state were used for the alignments. Results are shown for **varying hidden layers** for (a) the test set and (b) the test set.

4.3 Triphone Target Units

Number of Nodes	Number of Hidden Layers	Accuracy
500	3	73.36
400	5	73.09
500	5	73.7
650	5	74.32
400	7	73.31
500	7	73.73
650	7	74.09

Table 2: Recognition accuracy for DNN-HMMs with CD alignment with FBANK Init EDAZ input. Table summarizes results for different number of nodes per hidden layer and different number of hidden layers with a context shift of 4.



(a)

Figure 11: Performance accuracy with DNN-HMMs models for triphone target units. 500 nodes per hidden layer, context shift: 4, FBANK input features with 1st and 2nd differentials and 8 Gaussian components per state were used for the alignments. Results are shown for the test set with a **varying number of hidden layers**.

Monophones vs Triphones: While the performance improvements were not significant by augmenting the number of nodes and hidden layers with monophone target units, a larger improvement occurred when triphones were used as the DNN training labels. Overall, a better alignment leads to better results with DNN-HMMs.

Depth and Width: Small improvements were observed when augmenting the number of layers or nodes in the DNN structure. It is possible that using more hidden layers would improve further the accuracy, such gains are not expected to be significant. However, more training data needs to be available to achieve much higher accuracy.

4.4 Training and Decoding Time

CD-DNN-HMMs outperformed CD-GMM-HMMs in terms of recognition accuracy. However this gain was obtained at a significantly higher computational cost for training, primarily because the CD-DNN-HMM training algorithms are not easy to parallelize. Training a five-layer CD-DNN-HMM took significantly longer (at least 5 times) than CD-GMM-HMM even when the supplied GPU card was used. Stochastic gradient descend (SGD) algorithm was used to train the DNNs and SGD is inherently sequential and difficult to parallelize across machines [9] . The GPU can at least exploit the parallelism in the layered DNN structure.

To achieve higher accuracy, more training data is to be made available. The time spent on each epoch increases with more training data. However, as Dahl et al [9] suggest, fewer epochs will be needed when more training data is available. It is important to note that decoding with CD-GMM-HMM was still very efficient.

5 Biphone Models

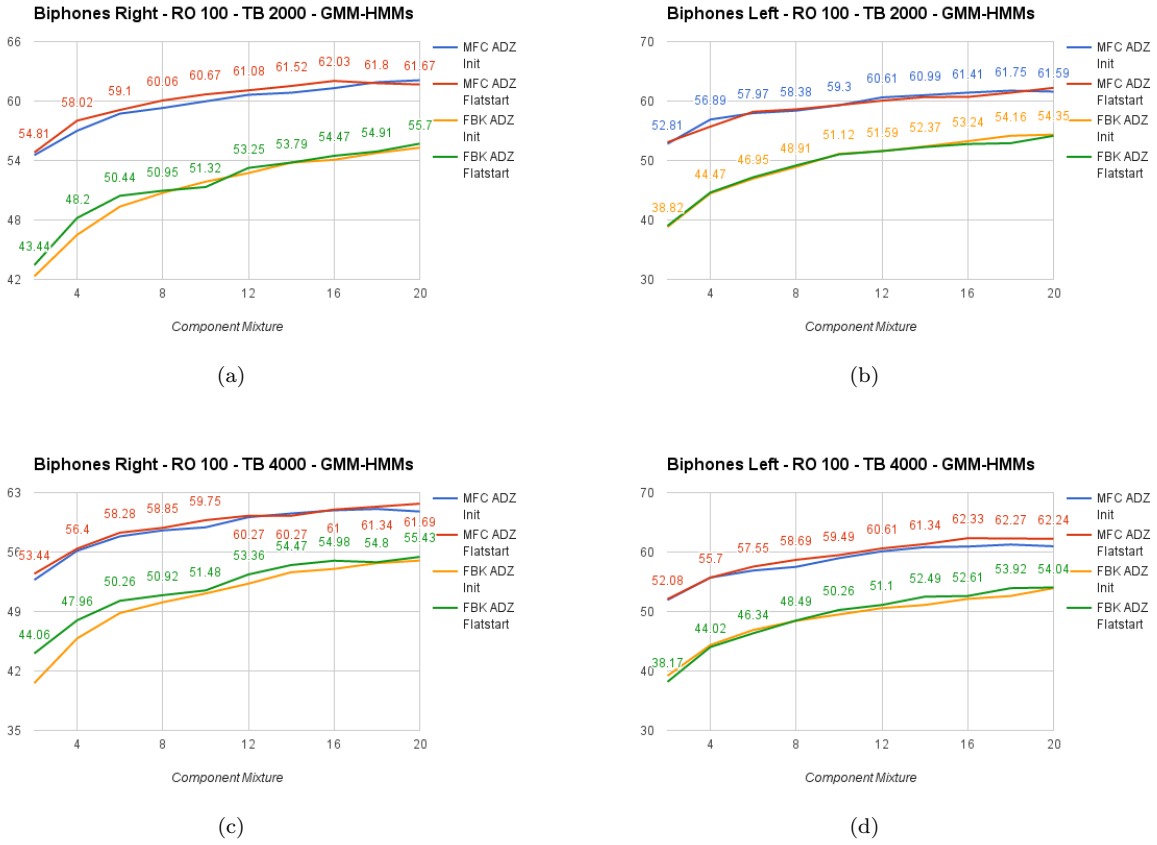
5.1 Biphone Models

In addition to training triphone models, biphone models were trained and tested. The scripts used are in Appendix A. Biphone label files and models were created using the provided monophone label files and models. Unclustered biphone models were then created and tying was performed using the same list of

clustering questions used for triphones. The tied clustered biphones models were then trained and tested with GMM-HMMs. The resulting state-level alignment of the training data was then used to train and test DNN-HMMs. Both left and right biphone clustering were implemented.

5.2 Front End Parametrization and Left vs Right Context

GMM-HMMs: As for monophones, and for both left and right context, training with a flatstart achieved higher performance than training after initialization with HInit. Training with MFCC input features performed significantly better than training with FBANK input features. Acoustic context without delta parameters were not considered. Increasing the number of mixtures increased accuracy. Decreasing the TB value for state tying did not necessarily improve accuracy. Differences between performances for left and right contexts are very small. The best accuracy was obtained with the left context for a flat start training with MFCC input and 20 Gaussians.



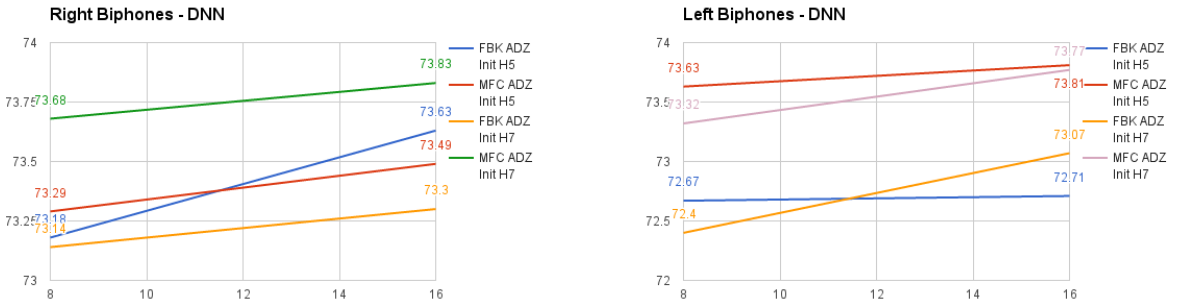
(a) (b)

(c) (d)

Figure 12: Performance accuracy with GMM-HMMs with a 2000 TB value with (a) right biphones and (b) left biphones and a 4000 TB with (c) right and (d) left biphones. Results are shown after state tying, for the test set with a RO value of 100.

DNN-HMMs: For both right and left biphones taget labels, differences in performances for different

input features were minimal. Using MFCC with 1st and 2nd differentials for the alignments achieved highest performance, which is consistent with the observation that better alignments lead to better results. Using 16 Gaussians per state instead of 8 for the alignment slightly increased accuracy. Overall, right biphone target units achieved better accuracy with DNN-HMMs.



(a) (b)

Figure 13: Performance accuracy with DNN-HMMs with (a) right biphones and (b) left biphones. Number of nodes: 650, context shift: 4, input features with 1st and 2nd differentials were used for alignments. Results are shown for the test set for 5 and 7 hidden layers, with alignments obtained from a GMM model with 8 and 16 Gaussians.

GMM vs DNN: The DNN-HMMs outperformed GMM-HMMs for both right and left contexts as for the triphone target units.

Comparison With Triphones: Training with triphones slightly performed better than biphones with both GMM-HMMs and DNN-HMMs. The slightly higher accuracy with triphones suggests that most of the contextual variation effect appears from a single one-side coarticulation. However, there is one other and important reason that explains the comparable performances between biphones and triphones: the final number of clustered states with biphones is lower than the one obtained with triphones, especially when same values of TB and RO are used. For N base phones, there are N^2 possible biphones and N^3 possible triphones and data sparseness becomes more significant with triphones. As seen in section 3.2.4 on the performance with respect to the number of clustered states and the number of Gaussians components, the recognition accuracy drops significantly when the number of clustered states increases above 1000 for models with more than 12 component mixtures. The training dataset is limited and does not allow a significant improvement with triphones compared to biphones.

6 Bigram Language Model

6.1 Setup

For the task of a large vocabulary continuous speech recognizer, the decoder tries to find the sequence of words W for a given input of acoustic vector Y. The problem is equivalent to $w = argmax_w\{P(w|Y)\} = argmax_w\{p(Y|w)P(w)\}$ where $p(Y|w)$ is determined by the acoustic model and the prior P(w) is determined by the language model. In the previous sections, a phone-loop grammar was used. In this section, we create and test a bigram language model.

The probability of a word sequence $W = w_1..w_K$ is

$$p(W) = \prod_{k-1}^{K} p(w_k|w_{k-1}, w_{k-2}, ..., w_1) \quad (1)$$

A bigram model assigns a probability to a word based on the previous word $P(w_k|w_{k-1})$. The bigram probabilities are estimated from the training set by counting occurrences to obtain ML estimates. One major setback with n-grams is data sparsity, which increases with increasing n-gram order. Such setback can be overcome by using discounting and backing-off. A phone-level back-off bigram language model is created using HLStats and context-dependent label files are used for training. The threshold count for including a bigram in a backed-off bigram language model was set to 3. The unigram floor probability was set to 1. A bigram network is build with HBuild and HVite. The scripts used are in Appendix B.

6.1.1 Grammar Scale

There is often a mismatch between the contribution of the acoustic model and the language model in speech recognisers. This is because the prior is estimated form a finite set of text documents and the likelihood is obtained from high dimensional observation densities [10]. We use a grammar scale to compensate for this mismatch. This scale raises the language model probability to the power of a constant.

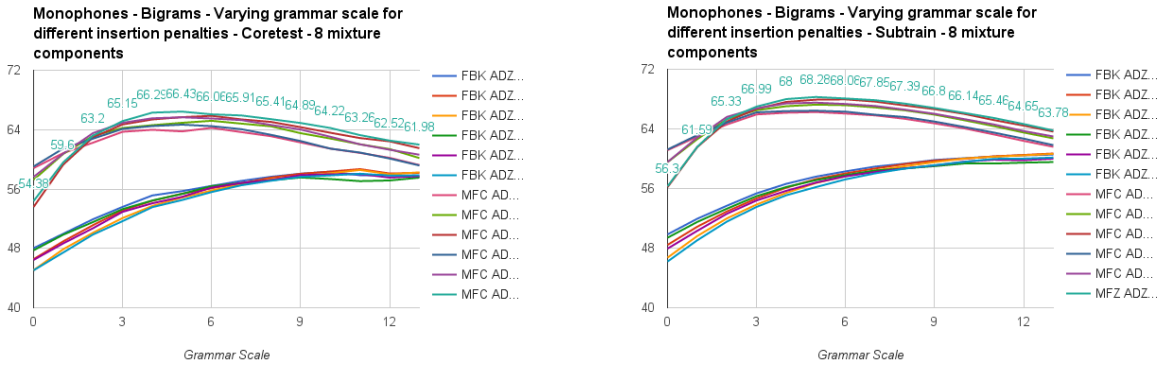
6.2 Experimental Findings

6.2.1 Grammar Scale and Insertion Penalty

Best results were obtained for a grammar scale of 3 and an insertion penalty of 0 for most models. Increasing the grammar scale further decreased accuracy. While it was expected to have better performance for lower insertion penalty, performance decreased when the insertion penalty decreased. Similar trends were observed for both the training and test set.

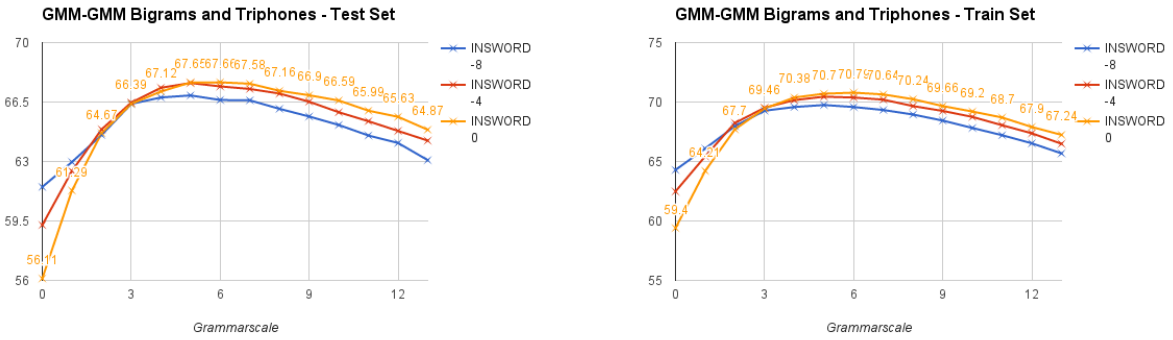
6.2.2 Monophones vs Triphones

6.2.3 GMM vs DNN



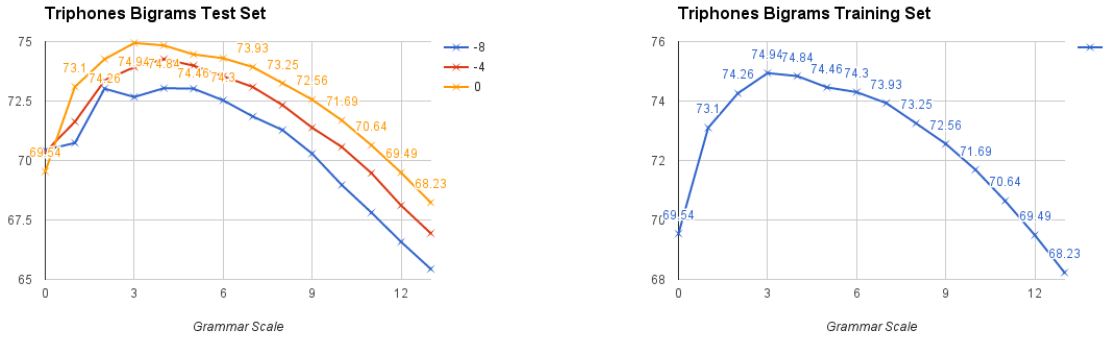
(a) (b)

Figure 14: Performance accuracy with GMM-HMMs models for an alignment with monophones. Acoustic context includes 1st and 2nd differentials. 8 Gaussian components per state were used and results are shown for (a) the test set and (b) the training set for different values of the **grammar scale** and the **insertion penalty**.



(a) (b)

Figure 15: Performance accuracy with GMM-HMMs models with triphones. Acoustic context includes 1st and 2nd differentials. 8 Gaussian components per state were used and results are shown for (a) the test set and (b) the training set for different values of the **grammar scale** and the **insertion penalty**. TB 5000 and RO 100 were used for state tying.



(a) (b)

Figure 16: Performance accuracy with DNN-HMMs models for an alignment with triphones. 500 nodes per hidden layer, context shift: 4, FBANK input features with 1st and 2nd differentials and 8 Gaussian components per state were used for the alignments. Results are shown for (a) the test set and (b) the training set for different values of the **grammar scale** and the **insertion penalty**.

7 Discussion and Model Comparison

Table 3 summarizes results obtained for all models considered in this practical.

Number of Gaussian Mixtures. With higher number of Gaussians, higher accuracy was obtained for all models, for both independent and dependent contexts. A higher number of Gaussians is able to model the asymmetric and multi-modal distributed data in speech. However, with increasing number of Gaussians, the number of parameters to estimate increases and the training data becomes insufficient to obtain robust parameter estimation. For DNN-HMMs, increasing the number of Gaussians for the models used for the alignments resulted in a larger number of parameters to estimate, increasing the run-time for training. Although better alignments lead to better results with DNN-HMMs, DNN models were all limited to a training with an alignment from GMM-HMM models with 8 Gaussians. In fact, using alignments with 16 Gaussians with biphones only led to an improvement in accuracy by 0.6% at most for DNN-HMMs.

Feature Input. Higher performance was obtained with MFCCs for the HMM-based model, since MFCCs are decorrelated which works better with Gaussian components that have diagonal covariance matrices. However, DNN-HMMs worked slightly better with the mel filter bank energies (FBANK), mainly because the Discreet Cosine Transform step to extract the MFCC features can lead to some information loss [11]. Augmenting feature vectors with derivatives improved accuracy for all models. Adding dynamic features to the input vectors compensate for the conditional independence assumption made by the HMM-based acoustic models [4].

Context. With GMM-HMMs, triphones significantly outperformed monophones but had slight improvement in accuracy over biphones. With GMM-DNNs, improvements when using context dependent alignments were not as significant as the ones observed with GMM-HMMs. Little value was added when using alignments with triphones, compared to biphones: Accuracy increased from 73.68% to 73.73% when alignments with triphones was used instead of biphones (right context), with a network of 500 nodes per hidden layer and 7 hidden layers.

When using context dependent phones, performance with GMM-HMMs depends on the final number of clustered states after the tying process, and the amount of available training data. As the number of clustered states increases, data sparsity increases and the training data becomes insufficient to obtain robust parameter

Table 3: Table summarizing results from all experiments carried in the practical.

	Model	8 components	16 components	Notes
GMM	Monophones FBANK	46.23	50.18	Flat Start
	Monophones MFCC	55.63	58.33	Flat start
	Monophones Bigram FBANK	55.71	59.02	GS 5 IS -8 Init
	Monophones Bigram MFCC	65.68	68.06	GS 5 IS 0 Init
	Biphones Left	58.38	61.41	MFC Init TB 2000 RO 100
	Biphones Right	59.3	61.3	MFC Init TB 2000 RO 100
	Triphones TB 5000	59.67	61.6	MFCC Init RO 100
	Triphones TB 1000	61.86	63.15	MFCC Init RO 100
	Triphones Bigrams	67.66	69.58	MFCC EDAZ INit TB 5000 RO 100 GS 6 IS
DNN	Monophones MFCC	68.72	70.02	H1 N500 Init EDAZ CS4
	Monophones FBANK	69.78	69.51	H1 N500 Init EDAZ CS4
	Monophones FBANK	72.52		H5 N650 Init EDAZ CS4
	Monophones MFCC	72.45		H5 N650 Init EDAZ CS4
	Monophones FBANK	72.67		H7 N500 Init EDAZ CS4
	Monophones MFCC	72.51		H7 N500 Init EDAZ CS4
	Biphones Left	73.32	73.77	MFCC EDAZ Init H7 N500
	Biphones Right	73.68	73.83	MFCC EDAZ Init H7 N500
	Triphones	73.7		FBANK Init EDAZ CS 4 N500 H5
	Triphones	74.32		FBANK Init EDAZ CS 4 N650 H5
	Triphones	73.73		FBANK Init EDAZ CS 4 N500 H7
	Triphones	74.09		FBANK Init EDAZ CS 4 N650 H7
	Triphones Bigrams	**74.94**		FBANK Init EDAZ CS 4 N500 H5 GS 3 IS 0
	Triphones Bigrams	74.61		FBANK Init EDAZ CS 4 N500 H7 GS 3 IS 0

estimation. With GMMs and with triphones, accuracy increased as the number of clusters increased up to 1000, and with a number of Gaussian mixtures below 14. Accuracy decreased as the number of states went above 1000. This decrease was more significant with GMM-HMMs with more than 14 Gaussian mixtures. Triphones only slightly performed better than biphones, mainly because data sparsity with biphones is less significant than with triphones.

Bigrams. The most significant improvement in recognition accuracy was observed when the bigram model was used instead of the phone-loop grammar. The accuracy increased from 55.63% to 65.68% when the bigram model was used with monophones and the GMM-HMM model with MFCC input features. DNN-HMMs with triphone target units and a bigram model achieved highest recognition accuracy of 74.94% among all experiments carried in this practical. This is mainly because the bigram model captures more local syntactic and semantic dependencies than the phone-loop grammar. While the bigram LM prevents modeling longer dependencies, increasing the n-gram order will increase data sparsity and little improvement in accuracy is expected.

DNN vs GMM. In general, DNN-HMMs provided significant improvements over GMM-HMMs for both CD and CI phone recognition. This gain is attributed to the concatenated features from the long context window used with DNNs [11]. This window consists of concatenated features from several consecutive speech frames, providing more context information to distinguish among different phones. Because of large overlaps in speech analysis windows, these frames are highly correlated and GMM-HMMs perform poorly with such highly correlated features (as seen with the FBANK input features). DNN-HMMs leverage highly correlated features and do not require dimensional reduction for decorrelation [12]. The posterior probabilities of all the states of all phonemes are simultaneously predicted; whereas with the GMM model, each phoneme is modeled separately from the rest. Training DNN-HMMs is quite expensive however, compared to training CD-GMM-HMMs. There is no meaningful interpretation of the DNN parameters and many effective adaptation techniques that have been developed and proven to work well for GMM-HMM systems cannot be easily applied to DNNs [12]. Decoding in CD-DNN-HMMs is very efficient and test time was not an issue.

8 Conclusion

This practical investigated the use of context dependent and context independent phone based large vocabulary speech recognition system. GMM-HMMs and DNN-HMMs models were used and refinements were explored.

When using HMMs, a lot of assumptions are made including that speech signals can be adequately represented by a sequence of spectrally derived feature vectors; that this sequence can be modeled using a HMM and that there is sufficient training data [4].

Adding dynamic features, increasing the number of Gaussian components and using context dependent phones increased phone recognition accuracy with GMM-HMMs.

Overall, using deep neural networks outperformed GMM models and most of this gain comes from exploiting information in neighboring frames. Using a bigram model further improved phone recognition and an accuracy of 75% was obtained with a DNN model with 5 hidden layers trained with an alignment obtained from a GMM model with triphones and 8 Gaussian components.

While the DNN-HMM system achieved highest accuracy, this gain was obtained at a significantly higher computational cost for training compared to GMM-HMMs. Selecting a good model involves a trade-off between training data availability, memory and run-time costs, and system complexity.

Improvements The input features and the alignments used for networks with 5 and 7 hidden layers were selected based on the single hidden layer experiments. It may be possible to obtain even better performance with the deeper models using a more exhaustive hyperparameter search. Using a better alignment to generate training labels for the DNN can also improve the accuracy, but such improvement might not be significant.

The models with DNNs were evaluated using a context width of up to 11 frames (CS from 1 to 5). In [7], the best performance was obtained with 17 frames. Further improvements could have been obtained with a larger context window.

References

[1] The HTK Book, version 3.4 (2006) by S. J. Young, G. Evermann, M. J. F. Gales, et al.

[2] Vertanen, K. (2007). "Baseline Wsj Acoustic Models for Htk and Sphinx: Training Recipes and Recognition Experiments.".

[3] Bhuvanagiri, Kiran Kumar, and Sunil Kumar Kopparapu. "Modified Mel Filter Bank to Compute MFCC of Subsampled Speech." arXiv preprint arXiv:1410.7382 (2014).

[4] Gales, Mark, and Steve Young. "The application of hidden Markov models in speech recognition." Foundations and trends in signal processing 1, no. 3 (2008): 195-304.

[5] Young, Steve J., Julian J. Odell, and Philip C. Woodland. "Tree-based state tying for high accuracy acoustic modelling." In Proceedings of the workshop on Human Language Technology, pp. 307-312. Association for Computational Linguistics, 1994.

[6] Chan, William, and Ian Lane. "Deep Recurrent Neural Networks for Acoustic Modelling." (2015).

[7] A. Mohamed, G. Dahl, G. Hinton, "Acoustic Modeling Using Deep Belief Networks", IEEE Transactions on Audio, Speech, and Language Processing, pp. 14 - 22, vol. 20, Jan. 2012

[8] Maas, Andrew L., Peng Qi, Ziang Xie, Awni Y. Hannun, Christopher T. Lengerich, Daniel Jurafsky, and Andrew Y. Ng. "Building DNN Acoustic Models for Large Vocabulary Speech Recognition." (2014).

[9] Dahl, G. E., Yu, D., Deng, L., & Acero, A. (2012). "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition". IEEE Transactions on Audio, Speech, and Language Processing, 20(1), 30-42.

[10] Rosti, Antti-Veikko Ilmari. "Linear Gaussian models for speech recognition." PhD diss., University of Cambridge, 2004.

[11] J. Pan, C. Liu, Z. Wang, Y. Hu, H. Jiang, "Investigation of Deep Neural Networks (DNN) for Large Vocabulary Continuous Speech Recognition: Why DNN Surpasses GMMs in Acoustic Modeling", International Symposium on Chinese Spoken Language Processing (ISCSLP), vol. 8, pp. 301 - 305, Dec. 2012

[12] Liu, S., and Sim, K. C. (2014). "On combining DNN and GMM with unsupervised speaker adaptation for robust automatic speech recognition". 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, 2014, pp. 195-199.

9 Appendix A : Scripts for the Biphone Models

9.1 step-xbir script for decoding HMMs with right biphones

```
#!/bin/tcsh
#$ -S /bin/tcsh

# Step1: tied cross-word biphone state GMM-HMMs training

set ALLARGS=($*)
set CHANGED
set NMIX = 1
set ROVAL = 200.0
set TBVAL = 200.0
set SRCDIR = $argv[1]
set HMMDIR = $argv[2]
set TGTDIR = $argv[3]

source $SRCDIR/environment
source /opt/intel/composerxe/bin/compilervars.csh intel64

mkdir -p $TGTDIR

# change to the target working dir and do the basic steups
cp $SRCDIR/environment $TGTDIR/environment
cd $TGTDIR
#     setup the tools dir
if ($?TOOLSDIR) then
    ln -s $TOOLSDIR ./tools
else
    ln -s $TIMITTOOLS ./tools
endif
if ( -d tools/htklib/cuda) then
    setenv LD_LIBRARY_PATH tools/htklib:$LD_LIBRARY_PATH
endif
#     setup the lib and cfgs
ln -s ${TIMITLIB} ./
cp lib/cfgs/basic${FEADIFF}.cfg ./basic.cfg
#     link the HMMs and HMM list
mkdir -p hmm0/work
ln -s $SRCDIR/$HMMDIR/MMF hmm0/work/MMF
ln -s $SRCDIR/hmms.mlist hmm0/work/hmms.mlist


# 1.1 setup the initial untied biphone HMMs (in hmm0) as a clone of monophone GMM-HMMs
echo "Step 1: Generate initial untied biphone GMM-HMMs
in hmm0/MMF and produce stats in hmm1/MMF"
#     generate the cross-word biphone labels
tools/htkbin/HLEd -A -D -V -T 1 -n hmm0/hmms.mlist -i train.mlf /home/ct506/MLSALT2/pracgmm/tools-b
echo 'CL hmm0/hmms.mlist' > hmm0/clone_xwbir.hed
#     generate the untied biphone HMMs
tools/htkbin/HHEd -A -D -V -T 1 -H hmm0/work/MMF -M hmm0 hmm0/clone_xwbir.hed hmm0/work/hmms.mlist
```

```
#     do single pass re-estimation
cp lib/cfgs/herest${FEADIFF}.cfg ./herest.cfg
cp lib/htefiles/HTE.xwtri hmm0/HTE.herest
echo "set TRAINDATALIST = lib/flists/train.scp" >> hmm0/HTE.herest      # training data file list
echo "set TRAINMLF = train.mlf" >> hmm0/HTE.herest                      # training labels
echo "set HMMLIST = hmm0/hmms.mlist" >> hmm0/HTE.herest                 # HMM list in hmm0/MMF
echo "set HECONFIG = herest.cfg" >> hmm0/HTE.herest                     # HERest config
echo "set BASICCONFIG = basic.cfg" >> hmm0/HTE.herest                   # basic feature config
echo "set HEBIN = tools/htkbin/HERest" >> hmm0/HTE.herest               # HERest binary file
#tools/scripts/herest hmm0/HTE.herest hmm0 hmm1
tools/scripts/hbuild hmm0/HTE.herest 1 2

# 1.2 do decision tree clustering to produce the tied biphone GMM-HMMs and do re-estimation
echo "Step 2: Do decision tree tying with hmm1/MMF and
hmm1/stats to produce hmm2/MMF and do re-estimation"
#     do decision tree based clustering
mkdir -p hmm10
set TREENAME = clustering_${ROVAL}_${TBVAL}.trees
set LISTNAME = clustering_${ROVAL}_${TBVAL}.mlist
set HEDNAME = clustering_${ROVAL}_${TBVAL}.hed
cat /home/ct506/MLSALT2/pracgmm/tools-bip/cluster_ROVAL_TBVAL_RB.hed |
sed "s=ROVAL=$ROVAL=g; s=TBVAL=$TBVAL=g; s=TREENAME=hmm10/$TREENAME=g;
s=LISTNAME=hmm10/$LISTNAME=g" > hmm10/$HEDNAME
tools/htkbin/HHEd -A -D -V -T 1 -H hmm1/MMF -M hmm10 hmm10/$HEDNAME hmm0/hmms.mlist > hmm10/LOG
ln -s $PWD/hmm10/$LISTNAME ./hmms.mlist
#     do re-estimation for multiple iterations
cp lib/htefiles/HTE.xwtri HTE.herest
echo "set TRAINDATALIST = lib/flists/train.scp" >> HTE.herest           # training data file list
echo "set TRAINMLF = train.mlf" >> HTE.herest                           # training labels
echo "set HMMLIST = hmms.mlist" >> HTE.herest                           # HMM list in hmm0/MMF
echo "set HECONFIG = herest.cfg" >> HTE.herest                          # HERest config
echo "set BASICCONFIG = basic.cfg" >> HTE.herest                        # basic feature config
echo "set HEBIN = tools/htkbin/HERest" >> HTE.herest                    # HERest binary file
tools/scripts/hbuild HTE.herest 11 14
#     reset variance floor
mkdir hmm14.0
echo "set HMMLIST = hmms.mlist" > hmm14.0/HTE.hhed
echo "set HHBIN = tools/htkbin/HHEd" >> hmm14.0/HTE.hhed
echo "LS hmm14/stats" > hmm14.0/varfloor.hed
echo "FA 0.1" >> hmm14.0/varfloor.hed
tools/scripts/hedit hmm14.0/HTE.hhed hmm14 hmm14.0 hmm14.0/varfloor.hed
ln -s $PWD/hmm14/stats hmm14.0/stats

# output the results
echo "Training complete: The final biphone GMM-HMM set is in "${lasthmm}
```

9.2 Script for producing the biphone list

```
indir = '/remote/mlsalt-2016/ct506/MLSALT2/pracgmm/convert/mfc13d/lib/mlists/mono+sil.mlist'

text = open(indir).read()

phones = text.split()
biphones_left = []
biphones_right = []
sil_right = []
sil_left = []
for phone1 in phones:
        for phone2 in phones:
                biphones_left.append(phone2+'-'+phone1)
                biphones_right.append(phone1+'+'+phone2)
for phone1 in phones:
        sil_right.append('RE sil sil+'+phone1)
        sil_left.append('RE sil '+phone1+'-sil')

with open('make-xwbir.led', 'wb') as brfile:
        brfile.write("\n".join(sil_right))

with open('make-xwbil.led', 'wb') as blfile:
        blfile.write("\n".join(sil_left))

with open('xwbileft+sil.mlist', 'wb') as blfile:
        blfile.write("\n".join(biphones_left))

with open('xwbiright+sil.mlist', 'wb') as rlfile:
        rlfile.write("\n".join(biphones_right))
```

10 Appendix B : Scripts for the Bigram Language Model

10.1 Commands for creating the bigram LM

```
# Create a phone-level back-off bigram language model
# -t Set the threshold count for including a bigram in a backed-off bigram language model.
../tools/htkbin/HLStats -A -D -T 1 -b back_off_bigram -t 3 -u 1.0 -o phone-list.txt \
../convert/mfc13d/lib/mlabs/train.mlf

# Build a bigram network
../tools/htkbin/HBuild -A -D -T 1  -n back_off_bigram phone-list.txt bigram_network
```

10.2 HTE file for the Bigram LM

```
#!/bin/tcsh

# HVite parameters
set HVTRACE = 1                 # tracing level
set HVPRUNE = 100.0             # -t
```

```
set HVGSCALE = 5.0              # -s
set HVIMPROB = 0.0              # -p
set HVNET = /home/ct506/MLSALT2/pracgmm/pracblm/bigram_network  # phone loop network
set HVVOC = /home/ct506/MLSALT2/pracgmm/pracblm/bigram-phones.dct       # dictionary

# HResults parameters
set HRTRACE = 1                 # tracing level
set WORDLIST = /home/ct506/MLSALT2/pracgmm/pracblm/phone-list.txt
set HREQSET = lib/eqsets/phone48_39.eqs # 39 phone scoring map
```