Investigation of Different Acoustic Modeling Techniques for Low Resource Indian Language Data

Conference Paper · March 2015

DOI: 10.1109/NCC.2015.7084860

3 authors, including:

Sriranjani Ramakrishnan
Indian Institute of Technology Madras
5 PUBLICATIONS **1** CITATION
SEE PROFILE

Murali Karthick Baskar
Brno University of Technology
7 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Investigation of Different Acoustic Modeling Techniques for Low Resource Indian Language Data

Sriranjani R[1], *Murali Karthick B*[2] *and Umesh S*[2]
Department of Electrical Engineering[1], Department of Applied Mechanics[2]
Indian Institute of Technology Madras, Chennai 600036, India
Email: 1) am12s036@smail.iitm.ac.in 2) {ee13s010 and umeshs}@ee.iitm.ac.in

***Abstract*—In this paper, we investigate the performance of deep neural network (DNN) and Subspace Gaussian mixture model (SGMM) in low-resource condition. Even though DNN outperforms SGMM and continuous density hidden Markov models (CDHMM) for high-resource data, it degrades in performance while modeling low-resource data. Our experimental results show that SGMM outperforms DNN for limited transcribed data. To resolve this problem in DNN, we propose to train DNN containing bottleneck layer in two stages: First stage involves extraction of bottleneck features. In second stage, the extracted bottleneck features from first stage are used to train DNN having bottleneck layer. All our experiments are performed using two Indian languages (Tamil & Hindi) in Mandi database. Our proposed method shows improved performance when compared to baseline SGMM and DNN models for limited training data.**

Index Terms: DNN, SGMM, low resource data, Indian languages, Hindi, Tamil, bottleneck

I. Introduction

In recent years, Automatic Speech Recognition (ASR) systems have shown significant improvement in performance due to new techniques based on DNN models. However for DNN model, large number of parameters are needed when compared to other acoustic modeling techniques [1]. Previous studies shows that DNN training is crucial with limited amount of data [2]. Thus building an effective acoustic model for a reliable ASR system becomes difficult especially when sufficient data are not available. In such situation, better data modeling techniques are needed in order to achieve good performance for low-resource data. SGMM [3] is one such technique to model low-resource data efficiently by estimating only less number of parameters.

In SGMM, the shared parameters are projected using state specific subspace vectors to obtain state specific GMM parameters [3]. The compact representation of SGMM leads to reduced number of parameters and provides robust estimation of model parameters even for limited training data.

The main motivation behind this paper is our recent efforts to build robust Indian language speech recognizer with limited transcribed data. This lead us to choose acoustic modeling techniques like SGMM and DNN for building better acoustic models. Although SGMM works well for less data our main focus is to improve the performance of DNN when compared to SGMM. This is due to the noise robust characteristics of DNN models [4].

Several methods has been proposed in DNN framework to obtain superior performance for less data. Dropout technique is one such method to avoid over-fitting and provide improvement [5]. Experiments performed using this method did not give substantial improvement over SGMM. Stacked bottleneck features extracted using low-rank matrix factorization are also used to improve DNN performance for low-resource speech recognition [6]. This lead us to propose a new approach in DNN framework for better modeling of less data using bottleneck features. In this paper languages are dealt separately, unlike cross-lingual and multilingual techniques which pool data [7]. Two stage DNN having a bottleneck layer is used in this approach. The bottleneck features of a language containing limited data is given as input to train a DNN containing bottleneck. The usage of bottleneck layer in two stages of DNN training works better for low-resource languages and shows a better improvement when compared with baseline SGMM and DNN models.

Experiments are performed using the proposed method for Hindi and Tamil language in Mandi database. Standard databases like RM and TIMIT are also used to substantiate our results. Our proposed method shows an average relative improvement of 8.8% and 11.2% over baseline SGMM system and relative improvement of 15.25% and 12.53% over baseline DNN model for Hindi and Tamil respectively.

This paper is organized as follows. Section II gives an overview of SGMM along with its importance in handling limited data. Section III explains the DNN methodology and training procedure with small datasets. In section IV, the experimental setup is given along with the details of parameters used. Section V provides an comparative analysis over the the performance of SGMM and DNN on Tamil, Hindi, RM and TIMIT. Finally, the two stage bottleneck method is explained in section VI.

II. SGMM

Recently, SGMM has been proposed to alleviate the problem of acoustic modeling when there is very limited training data, by exploiting the idea of that the mean of Gaussian components lie in a subspace. In this method the state dependent parameters (mean, covariance and mixture weights) are not estimated independently. Instead, we train a globally shared low dimensional subspace S from which the GMM models are trained. This captures the correlations between tied-states.

$\mathbf{M}_i$ is the mean subspace from which the mean of the GMM $\boldsymbol{\mu}_{ji}$ can be obtained using the state specific vector $\mathbf{v}_j$. The mixture weights ω_{ji} can be obtained from the weight vector $\mathbf{w}_i$ using $\mathbf{v}_j$. Each GMM has the same number of mixture components I for all states. The parameters $\mathbf{M}_i$, $\boldsymbol{\Sigma}_i$, $\mathbf{w}_i$ are shared across all the states ($\Sigma_{ji} = \Sigma_i$).

III. DNN

DNN is a multi-layer perceptron (MLP) which uses multiple set of frames as speech input to model the data. Pretraining is performed in DNN to learn the input at one layer at a time. Each of those layers are composed by a stack of probabilistic bipartite graphs known as Restricted Boltzmann Machine (RBM). The basic set of units that compound the RBM can be visible (**v**) units which fit the input data with a given distribution or hidden units (**h**) related with the learning process. Due to the fact that the features given as input are Gaussian, Gaussian-Bernoulli distribution is used at the first RBM. The remaining RBMs will use Bernoulli-Bernoulli distribution to model the data. In this paper, the training in the **DNN** is performed through the classification of input frames into tied-states via the Stochastic gradient descent (SGD) training method with a constant learning rate and a fixed mini batch size. The role of the RBM is related with the feature discrimination. In RBM, the inputs are shuffled in frame level as well as sentence level and sent into mini-batches. Finally, the obtained outputs from each RBM are used to initialize hidden layers in DNN which leads in a better classification of frames into tied-states.

IV. Experimental setup

The experiments are performed using Kaldi Speech recognition toolkit [8]. The DNN computations are done in NVIDIA Tesla M2070 GPU using CUDA. Intel Xeon x5675 with 24 CPUs and a CPU frequency of 3.07 GHz are used for the rest of simulations.

A. CDHMM

Preprocessing of input speech is done by 25ms width hamming window and 10ms shift. A number of 13 order Mel frequency cepstral coefficients (MFCC) are extracted and $\triangle + \triangle$ is applied. Linear discriminant analysis (LDA) is applied over the MFCC features. Cepstral mean variance normalization (CMVN) is done over these features. A maximum likelihood estimation (MLE) and maximum likelihood linear transform (MLLT) is performed over these normalized LDA features to obtain a baseline CDHMM. The parameter details of the CDHMM for different databases are given in table I.

Table I. % WER of baseline CDHMM system

Database	# Hours		# Ph	CDHMM	
	Train	Test		# Ts	#mix
RM	3.8	1.3	47	1449	8
TIMIT	3.1	0.1	38	1049	8
Tamil	1	4.3	39	225	4
	3			300	8
	5			750	6
	22			1250	14
Hindi	1	5.2	42	379	4
	3			454	4
	5			571	8
	22			1350	8

 # Ph - Number of phones, # Ts - Number of tied-states, #mix - Number of Gaussian mixtures

B. SGMM

The SGMM is built for each dataset by varying the mixtures as well as sub-states. In our method, the subspace dimension of 40 was fixed as an optimal value used for experiments. LDA features with 40 dimensions were used for training the SGMM. Initially an universal background model is built (UBM) with appropriate number of Gaussian mixtures for each dataset. The parameter details for SGMM training is shown in table II.

Table II. Parameter details for SGMM training

Database	No of Mixtures	No of Initial and final substates
TIMIT	400	2500 & 7000
RM	400	2500 & 7000
Mandi (1, 3 hours)	54	350 & 350
Mandi (5 hours)	128	450 & 450
Mandi (22 hours)	400	1500 & 1500

C. DNN

An experimental methodology has been implemented to perform the training in DNN: 90% of the train data is used as training set and the remaining 10 % is used as validation set. The DNN training for all datasets are performed using standard recipe as in Kaldi [8]. Secondly, speaker adapted features are obtained by applying Linear discriminant analysis (LDA) in combination with Maximum likelihood linear transform (MLLT) is applied over the CDHMM model for pre-training. Splicing is done over these features with 9 frames before and after each center frame. In our work we performed pre-training by stacking Restricted Boltzmann machine's (RBM). This process is followed by initializing layers of the DNN using the weights obtained from the RBMs in a reverse manner.

Table III. Parameter details for DNN training

Parameter	Value
(a) Training data ($< 5\ hours$)	
Hidden layers	5
Hidden Units	1024
(b) Training data ($> 5\ hours$)	
Hidden layers	6
Hidden Units	2048
Learning rate	0.008
Fixed minibatch size	250
Momentum or regularization	Nil
(c) Dropout parameters	
Initial Dropout scale	0.9
Final Dropout scale	0.9

The DNN is trained discriminatively to optimize the weights and then classify frames into tied-states. The initial number of hidden layers is kept as 2 and the number of hidden layers are increased during training. The learning rate is reduced by halving after iteration if the frame classification accuracy does not improve. The parameters used for DNN training is mentioned in table III. These parameters are fixed as per Kaldi's DNN implementation. Finally, an affine transform is applied at each layer to do de-correlation. Softmax component is applied at the final layer to normalize the output probabilities of each tied-state.

Dropout method is performed only during the DNN training phase and implemented using Kaldi+PDNN [9]. Randomly some hidden nodes are selected based on a comparison between the probability and a dropout factor. The initial and

final dropout scales are chosen empirically and mentioned in table III

Database used:

The **TIMIT** database [10] is recorded on eight principal dialects of American English for 490 speakers with a sampling rate of 16 kHz. From each speaker 8 utterances were recorded. The speakers were divided into two sets with 462 speakers for training and 28 speakers for testing. The **RM Corpus [11]** with sampling rate of 16KHz, is recorded with 168 speakers. We use 109 speakers for training and 59 for testing. **Mandi Database** has been selected to perform the Indian language experiments. The data was collected for building Automatic IVR system in order to get the price of Agricultural commodities in Indian languages. This database comprises six of the major languages in India: Tamil, Telugu, Hindi, Bengali, Assamese and Marathi . The full dataset were collected among various states of India. Among these six languages, only Tamil and Hindi have been chosen for our experiments. All the recorded data is saved in RIFF (little-endian) format as well as WAVE audio, Microsoft PCM, 16 bit and mono with a sampling rate of 8 KHz. As the database was recorded on an rural environment, the speakers were selected among farmers. In our experiments the full databases have been separated into 1 hour, 2 hours, 3 hours, 5 hours and 22 hours sets to show the effects of different acoustic modeling techniques in terms of the amount of training data.

V. Analysis of performance for different acoustic models

The results are analyzed for different acoustic models in two conditions: sufficient data and limited data. The performance is compared for all 3 datasets, RM, TIMIT and Mandi data - Hindi and Tamil languages.

Table IV. Performance of datasets and parameter count for sufficient data (>3 hours) using CDHMM, SGMM and DNN

Datasets	CDHMM	SGMM			DNN	
		% WER	#ps	#Ts	%WER	#ps
TIMIT (3.8 hrs)	28.38	21.60	*1.0*	*2526*	20.22 *	*4.7*
RM (4 hrs)	3.41	1.71	*1.1*	*4363*	1.58	*7.2*
Hindi (5 hrs)	9.10	6.79	*0.3*	*540*	5.96	*10.3*
Tamil (5 hrs)	27.62	22.56	*0.6*	*747*	22.50	*9.0*
Hindi (22 hrs)	6.91	4.62	*1.2*	*6417*	3.91	*16.1*
Tamil (22 hrs)	21.64	20.31	*0.9*	*1155*	20.01	*26.1*

#ps - Number of parameters in millions, #Ts - Number of tied states, hrs - hours
** - Obtained using DNN dropout method*

Performance comparison for sufficient data (Training data > 3 hrs):

As shown in Table IV, for training data more than 3hrs, the DNN gives improved performance across all the datasets. SGMM shows 4.4 % relative improvement over DNN for TIMIT while for RM performance of SGMM is comparable to DNN. DNN dropout method showed improvement over DNN for TIMIT. On the whole, for sufficient data, DNN outperform CDHMM and SGMM methods.

Table V. Performance of datasets with limited data ($\leq$ 3 hours) using CDHMM, SGMM and DNN

Datasets	CDHMM	SGMM			DNN	
		%WER	#ps	#Ts	%WER	#ps
Hindi (1 hr)	15.76	13.66	0.2	387	14.02	7.7
Tamil (1 hr)	43.73	34.50	0.1	604	37.10	7.5
Hindi (3 hr)	11.59	10.00	0.3	395	11.48 *	10.0
Tamil (3 hr)	31.00	27.10	0.2	1703	28.46	10.0

#ps - Number of parameters in millions, #Ts - Number of tied states
** - Obtained using DNN dropout method*

Performance comparison for less data (Training data $\leq$ 3 hrs):

SGMM shows a relative improvement of 2.56%, 7.00%, 12.89% and 4.77% over DNN for Hindi (1hr), Tamil (1hr), Hindi (3hrs) and Tamil (3hrs) respectively. SGMM outperform DNN in all these cases. This is due to less number of parameters used by SGMM compared to DNN. For example in table V, for Tamil (1hr) SGMM uses 0.1 million parameters compared to 7.5 million parameters used by DNN. Owing to the model complexity, the available data is limited for training DNN. This leads to over fitting of the parameters and consequently a drop in its performance.

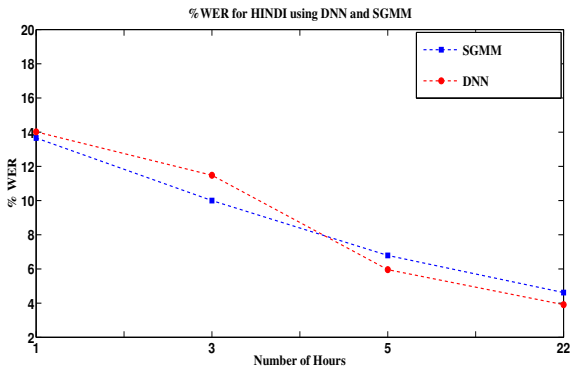


Figure 1. Variation in performance of DNN and SGMM in terms of % WER for different amount of training input using Hindi data

Another important factor of SGMM is that the amount of time taken to train is less than DNN. It takes approximately 30 minutes to train SGMM for Tamil (1hr) while DNN takes 4 hours in our set up as mentioned in section III.

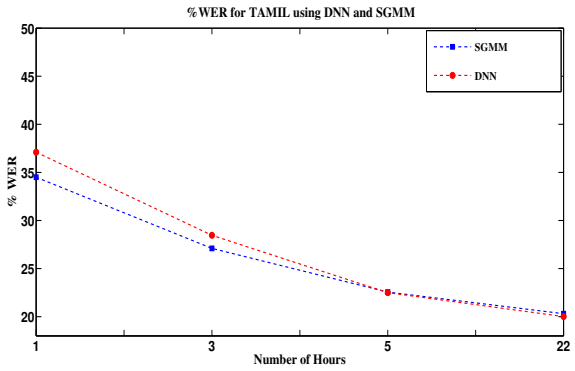


Figure 2. Variation in performance of DNN and SGMM in terms of % WER for different amount of training input using Tamil data

Performance comparison for Hindi and Tamil data for SGMM and DNN:

Experiments were performed with different amount of data for Tamil and Hindi. Figures 1 and 2 shows performance of SGMM and DNN for 1, 3, 5 and 22 hours of Tamil and Hindi data. We observe that as the amount of data decreases, the performance of SGMM starts to increase over DNN. This concludes that for low resource DNN does not give good performance. In order to improve the performance, a new approach is proposed in section VI.

VI. Two-stage Bottleneck Method

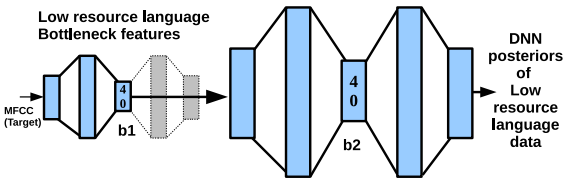
The main objective of this method is to avoid feature components which are irrelevant for modeling the low resource data. This helps in removing unwanted features to be carried to the output layer. A better discrimination between classes can be achieved through a bottleneck layer in a DNN due to reduction of input dimensions. This idea has been exploited in this method by passing the output of a first bottleneck layer into another DNN containing a bottleneck. Now, feature components are well refined and discriminated due to the placement of bottleneck at the second stage. When a low resource data is used as input in this framework better performance of the DNN for limited data increases significantly

The implementation of this model is done in the following steps.

Step 1: A 5 layer DNN having a bottleneck layer $b1$ is trained with a low-resource data. The context-dependent phone alignments are used as class labels during DNN training The bottleneck layer $b1$ of 40 dimensions compresses the input features and learns the less dimensional representation of data to perform improved classification. After training all layers the bottleneck features are extracted by discarding the hidden layer and output layer.

Step 2: The bottleneck features extracted from $b1$ are sent to a single 5 layer DNN with bottleneck $b2$ of 40 dimensions. The 5 layers are trained to obtain context-dependent class posteriors at the output layer. The bottleneck $b2$ acts as a classifier with less number of nodes for doing compressed classification of inputs.

Figure 3. Schematic Block Diagram of the Two Stage Bottleneck DNN system



Step 3: Classified posterior probabilities are obtained from the final layer of DNN.

In this method the added advantage comes by using bottleneck features for training DNN and using a second bottleneck $b2$ for obtaining further efficiency in classification.

VII. Experiments and Results

Experiments are performed on one hour data (low resource data) of Hindi and Tamil. The training data is split into 90% for training and 10% for validation. The proposed system is trained as shown in section VI. To illustrate the training procedure, consider Tamil (1 hour) dataset for which LDA features are given as input in the first stage to extract bottleneck features. These bottleneck features are given as input to DNN containing bottleneck and the posterior probabilities are obtained at the output layer. Each train set is tested with the corresponding target language.

From table V it was shown that SGMM outperforms DNN for less data and in this proposed approach we improve the DNN performance over SGMM. Table VI compares the performance of DNN with bottleneck architecture using bottleneck features **(Bnk-Bnk)** and the SGMM system. The proposed system shows consistent improvement compared with the SGMM and DNN based models when trained with low-resource data. A relative improvement of 4.4% and 13.2% is obtained for Hindi (1hr) and Hindi (3hr) respectively. In case of Tamil language a relative improvement of 4.9% and 9.2% is obtained for Tamil (1hr) and Tamil (3hr) respectively. Thus the proposed system can be used for modeling less data efficiently.

Table VI. Performance of Two-stage bottleneck system

Datasets	% WER			%Rel Imp. of Bnk-Bnk wrt SGMM	%Rel Imp. of Bnk-Bnk wrt DNN
	SGMM	DNN	Bnk-Bnk		
Hindi (1hr)	13.66	14.02	13.08	4.4	6.7
Tamil (1hr)	34.5	37.10	32.8	4.9	11.5
Hindi (3hr)	10.00	11.48	8.68	13.2	23.8
Tamil (3hr)	27.10	28.46	24.6	9.2	13.56

% Rel Imp. - % Relative Improvement, wrt - with respect to

VIII. Conclusions and Future Work

This paper investigates the effect of varying amounts of training data on different acoustic modeling techniques namely, CDHMM, SGMM and DNN. The experiments were performed on two standard datasets: TIMIT & RM and on Mandi dataset for two Indian languages viz, Hindi and Tamil. Our analysis shows that for sufficient training data (>3hours), DNN outperform SGMM and CDHMM, while for less data, SGMM outperform DNN. To improve the performance of DNN for less data, a varied approach based on two stage bottleneck system has been proposed. The proposed system gives consistent improvement for Indian languages in which only limited amount of transcribed data is available.

IX. Acknowledgements

This research was supported in part by the Technology Development in Indian language programme of the Ministry of Communication & Information Technology, Govt. of India. The DIT-ASR consortium includes IIT Madras, IIT Bombay, IIT Guwahati, IIT Kanpur, IIIT Hyderabad, Tata Institute of Fundamental Research Mumbai and Center for Development of Advanced Computing Kolkata. The authors are grateful to all the members of Consortium for collecting and transcribing agricultural Commodity data.

REFERENCES

[1] D. Yu, F. Seide, G. Li, and L. Deng, "Exploiting sparseness in deep neural networks for large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pp. 4409–4412, IEEE, 2012.

[2] P. Swietojanski, A. Ghoshal, and S. Renals, "Unsupervised cross-lingual knowledge transfer in dnn-based lvcsr.," in *SLT*, pp. 246–251, 2012.

[3] D. Povey, L. Burget, M. Agarwal, P. Akyazi, F. Kai, A. Ghoshal, O. Glembek, N. Goel, M. Karafiát, A. Rastrow, *et al.*, "The subspace gaussian mixture model a structured model for speech recognition," *Computer Speech and Language*, vol. 25, no. 2, pp. 404–439, 2011.

[4] M. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 7398–7402, May 2013.

[5] Y. Miao and F. Metze, "Improving low-resource cd-dnn hmm using dropout and multilingual dnn training," in *Proc. Interspeech*, pp. 2237–2241, 2013.

[6] Y. Zhang, E. Chuangsuwanich, and J. Glass, "Extracting deep neural network bottleneck features using low-rank matrix factorization,"

[7] S. Thomas, S. Ganapathy, and H. Hermansky, "Cross-lingual and multi-stream posterior features for low resource lvcsr systems.," in *INTERSPEECH*, pp. 877–880, 2010.

[8] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, *et al.*, "The kaldi speech recognition toolkit," in *Proc. ASRU*, pp. 1–4, 2011.

[9] Y. Miao, "Kaldi+ pdnn: Building dnn-based asr systems with kaldi and pdnn," *arXiv preprint arXiv:1401.6984*, 2014.

[10] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon Technical Report N*, vol. 93, p. 27403, 1993.

[11] D. Pallett, J. Fiscus, and J. Garofolo, "Resource management corpus: September 1992 test set benchmark test results," in *Proceedings of ARPA Microelectronics Technology Office Continuous Speech Recognition Workshop, Stanford, CA*, 1992.