Ensemble of Gaussian Mixture Localized Neural Networks with Application to Phone Recognition

Ruchir Travadi, Shrikanth Narayanan

Signal Analysis and Interpretation Lab,
University of Southern California, Los Angeles - 90089

travadi@usc.edu, shri@sipi.usc.edu

Abstract

In this paper we present Ensemble of Gaussian Mixture Localized Neural Networks (EGMLNNs), a model for the joint probability density of input as well as output variables of any general mapping to be estimated. The model aims at identifying clusters in the input data, thereby replacing one complex classifier with an ensemble of relatively simpler classifiers, each of which is localized to operate within its associated cluster. We present an algorithm for maximum likelihood parameter estimation for this model using Expectation Maximization (EM). The reported results on phone recognition task on TIMIT database show that the model is able to obtain performance improvement over a single complex classifier while also reducing the computational complexity required for testing.

Index Terms: Neural Networks, Acoustic Modeling, Speech Recognition, Phone Recognition

1. Introduction

In the past few years, Deep Neural Networks (DNNs) have emerged as one of the most successful machine learning tools across a variety of tasks [1–3]. Advances in the machine learning community are also being accompanied by vast amounts of data that is being made available for training these systems.

As a result it has become possible to train huge networks that are able to show superior generalization performance. However, this trend poses a challenge in terms of the associated computational requirements for training and testing these systems, even with present day computing abilities. More recently, ensembles of deep networks have been gaining popularity [4], raising the computational requirements even further.

In view of these developments, our objective in this work is to design an ensemble composed of relatively small neural networks, that is able to outperform a single big network, while reducing the associated computational burden.

Our approach towards designing such an ensemble involves incorporating the estimation of input density into the model. More specifically, we propose a probabilistic model that jointly estimates clusters in the input space, as well as the behaviour of the mapping from the input to the output locally in each cluster. Our hypothesis behind making such a choice for the model is that if the input space is clustered, we expect the behaviour of the mapping to be smoothly varying within a cluster, but independent across widely separated clusters. This is a reasonable assumption for most real world applications and is also a key implicit assumption in most regularization techniques.

Under this assumption, we can view one complex learning problem alternatively as a collection of simpler, cluster-localized learning problems. In terms of model training, this translates into replacing one complex parameter estimation task with a number of simpler, parallelizable tasks. Similarly, while testing, cluster associations can be obtained from input with a relatively modest computational requirement. Once they have been computed, we only need to invoke a small number of simpler local classifiers rather than a complex global classifier.

From a historical perspective, this is similar in philosophy to the Mixture of Experts (ME) approach [5], where a gating network is used to partition the input space, and an ensemble is trained so that individual networks are active within localized regions. However, the partitioning of the input space in the ME approach is based on the degree of success of different classifiers, and the focus is on finding a loss function that promotes the gating network outputs to be concentrated on a small subset of the ensemble. In contrast, the analog of gating function in our case happens to depend on the density of the input space, and the likelihood of observed data naturally acts as an objective that satisfies the desired loss function properties in ME.

The joint density model we propose is very similar to the alternative ME approach adopted in [6]. However, the focus in [6] is on eliminating the iteration involved in Maximization step of EM algorithm by using simpler expert networks for which the problem is analytically solvable. Instead, our goal is to divide a complex modeling task into a set of simpler, localized subtasks, without compromising on model capacity. Therefore, we use DNNs as expert networks that, unlike the models in [6], require a few iterations of backpropagation, but offer a significantly greater modeling complexity.

In the context of acoustic modeling for speech recognition as an application, as we later point out in Section 2, our model could be viewed as a generalization of both Gaussian Mixture Model (GMM) based as well as Deep Neural Network (DNN) based approaches to the problem.

The remainder of this paper is organized as follows: In Section 2 we explain the theory and algorithms related to our proposed model. In Section 3, we describe the experimental setup for phone recognition on TIMIT database [7]. The results are then reported in Section 4 followed by conclusion.

2. Ensemble of Gaussian Mixture Localized Neural Networks

Let $\boldsymbol{x}$ and $\boldsymbol{y}$ denote, respectively, the input and output variables of a mapping $\boldsymbol{y} = f(\boldsymbol{x})$. Ensemble of Gaussian Mixture Localized Neural Networks (EGMLNN) is a model for the joint density of variables $\boldsymbol{x}$ and $\boldsymbol{y}$, with the following distribution:

$$p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\Omega}) = \sum_{c=1}^{C} p_c\, p(\boldsymbol{x}|c, \boldsymbol{\theta_c})\, p(\boldsymbol{y}|\boldsymbol{x}, c, \mathcal{W}_c) \qquad (1)$$

where $\mathbf{\Omega} = \{p_c, \boldsymbol{\theta_c}, \mathcal{W}_c\}_{c=1}^{C}$ denotes the parameter space of the model. Here, the index variable c represents a particular cluster, p_c denotes the prior probability of selecting cluster c, $p(\boldsymbol{x}|c, \boldsymbol{\theta_c})$ denotes the input variable density associated with cluster c, and $p(\boldsymbol{y}|\boldsymbol{x}, c, \mathcal{W}_c)$ denotes the density associated with output variable $\boldsymbol{y}$ conditioned on the cluster c and the input $\boldsymbol{x}$.

Gaussian Mixture Models (GMMs) are a widely popular choice as a model for representing clusters in the input feature space. Hence, in this paper, we model the cluster-specific input density as a multidimensional Gaussian: $p(\boldsymbol{x}|c, \boldsymbol{\theta_c}) \sim \mathcal{N}(\boldsymbol{\mu_c}, \boldsymbol{\Sigma_c})$. It follows from this assumption that the corresponding density of the input variable $\boldsymbol{x}$ is a Gaussian mixture:

$$p(\boldsymbol{x}|\mathbf{\Omega}) = \sum_{c=1}^{C} p_c \, p(\boldsymbol{x}|c, \boldsymbol{\theta_c}) \tag{2}$$

Similarly, Neural Networks are among the most popular models for function estimation, particularly given their recent success on a range of different tasks. Hence, we assume that the density $p(\boldsymbol{y}|\boldsymbol{x}, c, \mathcal{W}_c)$ is the output of a neural network with input $\boldsymbol{x}$, weights $\mathcal{W}_c$, and a softmax output layer.

Under these assumptions, it follows that the overall density $p(\boldsymbol{y}|\boldsymbol{x})$ of the output $\boldsymbol{y}$ conditioned on the input $\boldsymbol{x}$ is a convex combination of cluster-specific densities $p(\boldsymbol{y}|\boldsymbol{x}, c)$ with the Gaussian mixture posteriors acting as convex weights:

$$p(\boldsymbol{y}|\boldsymbol{x}, \mathbf{\Omega}) = \sum_{c=1}^{C} p(c|\boldsymbol{x}, \boldsymbol{\theta_c}) p(\boldsymbol{y}|\boldsymbol{x}, c, \mathcal{W}_c) \tag{3}$$

In this context, with speech acoustic modeling as an application in mind, we would like to point out an interesting connection of EGMLNNs to existing approaches in this domain.

Traditionally, GMMs have been used as tools to model the class-conditional densities $p(\boldsymbol{x}|\boldsymbol{y})$ for acoustic modeling. If we assume in our model that the density $p(\boldsymbol{y}|\boldsymbol{x}, c)$ is independent of the input variable $\boldsymbol{x}$, i.e. $p(\boldsymbol{y}|\boldsymbol{x}, c) = p(\boldsymbol{y}|c)$ for all components c, and further that each component is associated with just one label, i.e. $p(\boldsymbol{y}|c) = 1$ if component c is associated with label $\boldsymbol{y}$ and 0 otherwise, then the EGMLNN model reduces to the traditional GMM approach:

$$p(\boldsymbol{x}|\boldsymbol{y}, \mathbf{\Omega}) = \frac{1}{p(\boldsymbol{y})} \sum_{\substack{c=1 \\ p(\boldsymbol{y}|c)=1}}^{C} p_c \, p(\boldsymbol{x}|c, \boldsymbol{\theta_c}) \tag{4}$$

where $p(\boldsymbol{y}) = \sum_{c=1}^{C} p_c \, p(\boldsymbol{y}|c)$ is a constant.

On the other hand, if we assume that the Neural Network weights $\mathcal{W}_c$ are tied across clusters, i.e. $\mathcal{W}_c = \mathcal{W}$ for all c, then it follows that $p(\boldsymbol{y}|\boldsymbol{x}, c, \mathcal{W}_c) = p(\boldsymbol{y}|\boldsymbol{x}, \mathcal{W})$, and the model reduces to a single neural network representing $p(\boldsymbol{y}|\boldsymbol{x})$, while the parameters p_c and $\boldsymbol{\theta_c}$ only represent the density on the input space $p(\boldsymbol{x})$ which plays no role in determining $p(\boldsymbol{y}|\boldsymbol{x})$.

Therefore, in the context of acoustic modeling, the EGMLNN model can be viewed as a generalization under which both the traditional GMM approach and Neural Network based approaches could be realized as special cases.

In this paper, as an initial effort, we concentrate on phone recognition as an application for our model. In the remainder of this section, we summarize how the EGMLNN model fits into the overall scheme of a phone recognition system, introduce some of the associated notation, and present the theory associated with training as well as testing the model.

2.1. EGMLNN for Phone Recognition

Let $\boldsymbol{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_T}\}$ denote a sequence of frames consisting of speech feature vectors for an utterance. The task of phone recognition involves obtaining the corresponding sequence of phones $\{\boldsymbol{p_1}, \boldsymbol{p_2}, \ldots, \boldsymbol{p_M}\}$. Typical approaches for solving this task involve training a Hidden Markov Model (HMM) for modeling transitions across phonetic states, and using either Gaussian mixtures to model the density of acoustic feature vector given the state, or using a neural network to model state posteriors given the feature vector. We use the latter approach, while replacing a single neural network with the EGMLNN model. Therefore, in a typical setup using EGMLNN, the HMM states act as the output variables $\boldsymbol{y}$. However, as later mentioned in Section 3, we use a setup involving HMM states with tied parameters, where a tied set of HMM states is identified by a unique index known as a pdf-id. Therefore, in our actual experimental setup, pdf-ids that represent a set of tied HMM states act as output variables.

Let $\boldsymbol{Y} = \{\boldsymbol{y_1}, \boldsymbol{y_2}, \ldots, \boldsymbol{y_T}\}$ denote the sequence of pdf-ids associated with every frame. Let $\boldsymbol{w_t}$ denote a window of $2K + 1$ frames centered at t: $\boldsymbol{w_t} = \{\boldsymbol{x_{t-K}}, \boldsymbol{x_{t-K+1}}, \ldots, \boldsymbol{x_t}, \ldots, \boldsymbol{x_{t+K-1}}, \boldsymbol{x_{t+K}}\}$. Then, the joint distribution for the data $(\boldsymbol{X}, \boldsymbol{Y})$ according to the EGMLNN model is given as:

$$p\,(\boldsymbol{X}, \boldsymbol{Y}) = \prod_{t=1}^{T} \sum_{c=1}^{C} p_c \, p(\boldsymbol{x_t}|c, \boldsymbol{\theta_c}) \, p(\boldsymbol{y_t}|\boldsymbol{w_t}, c, \mathcal{W}_c) \tag{5}$$

Note the subtle difference in the density described here as compared to that in Equation (1): The input to the neural network consists of a window of frames rather than a single frame. This is typical of most DNN based systems for acoustic modeling (e.g. [8]). However, the input density modeling is still based on a single frame as GMMs are rather sensitive to dimensionality of the input.

2.2. Model Training

Parameters for this model can be estimated using Maximum Likelihood Estimation. However, direct maximization of the likelihood function happens to be intractable in this case, so we resort to the Expectation-Maximization (EM). Let c_t denote the mixture component associated with frame t. We treat these variables $\boldsymbol{c} = \{c_1, c_2, \ldots, c_T\}$ as hidden variables.

2.2.1. E-step

Let $\mathbf{\Omega}$ denote the current parameter estimates. It can be shown that the posterior probability of the component associations taking the values $\{c_1, c_2, \ldots, c_T\}$ is given as below:

$$p(c_1, \ldots, c_T|\boldsymbol{X}, \boldsymbol{Y}, \mathbf{\Omega}) = \prod_{t=1}^{T} \gamma_{t,c_t}, \text{ where}$$

$$\gamma_{t,c_t} = \frac{p_{c_t} p(\boldsymbol{x_t}|c_t, \boldsymbol{\theta_{c_t}}) p(\boldsymbol{y_t}|\boldsymbol{w_t}, c_t, \mathcal{W}_{c_t})}{\sum\limits_{c_t=1}^{C} p_{c_t} p(\boldsymbol{x_t}|c_t, \boldsymbol{\theta_{c_t}}) p(\boldsymbol{y_t}|\boldsymbol{w_t}, c_t, \mathcal{W}_{c_t}))} \tag{6}$$

Here, γ_{t,c_t} is the posterior probability of mixture component c_t being associated with frame t

2.2.2. M-step

Let $\mathbf{\Omega^*}$ denote the new parameters to be estimated. By algebraic manipulation, the expected log-likelihood function to be maximized with respect to $\mathbf{\Omega}^*$ could be simplified to:

$$\mathbb{E}_{p(\boldsymbol{c}|\boldsymbol{X},\boldsymbol{Y})} \log p\left(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{c}|\mathbf{\Omega}^*\right) = \sum_{c=1}^{C}\sum_{t=1}^{T} \gamma_{t,c} \log p_c^*$$
$$+ \sum_{c=1}^{C}\sum_{t=1}^{T} \gamma_{t,c} \log p\left(\boldsymbol{x_t}|c, \boldsymbol{\theta_c^*}\right)$$
$$+ \sum_{c=1}^{C}\sum_{t=1}^{T} \gamma_{t,c} \log p\left(\boldsymbol{y_t}|\boldsymbol{w_t}, c, \mathcal{W}_c^*\right) \qquad (7)$$

The corresponding parameter updates to maximize the expected log-likelihood are given as below:

$$p_c^* = \frac{1}{T}\sum_{t=1}^{T} \gamma_{t,c}, \quad \boldsymbol{\mu_c^*} = \frac{1}{p_c^*}\sum_{t=1}^{T} \gamma_{t,c}\, \boldsymbol{x_t}$$
$$\mathbf{\Sigma_c^*} = \frac{1}{p_c^*}\sum_{t=1}^{T} \gamma_{t,c}\, \left(\boldsymbol{x_t} - \boldsymbol{\mu_c^*}\right)\left(\boldsymbol{x_t} - \boldsymbol{\mu_c^*}\right)^T \qquad (8)$$

Maximizing the term involving $\mathcal{W}_c^*$ in equation (7) is equivalent to minimizing the cross-entropy between actual labels $\boldsymbol{y_t}$ and the corresponding Neural Network outputs for component c, with component posteriors $\gamma_{t,c}$ acting as frame weights The updated parameters $\{\mathcal{W}_c^*\}_{c=1}^{C}$ can thus obtained by performing C independent frame-weighted backpropagation updates which can potentially be carried out in parallel. This flexibility of dividing the training process into smaller, potentially parallelizable tasks is one major advantage of the EGMLNN model with relatively simpler DNNs against a single, complex DNN model.

2.3. Testing

For testing, component posteriors need be obtained as below:

$$p(c_t|\boldsymbol{x_t}, \mathbf{\Omega}) = \frac{p_{c_t} p(\boldsymbol{x_t}|c_t, \boldsymbol{\theta_t})}{\sum_{c_t=1}^{C} p_{c_t} p(\boldsymbol{x_t}|c_t, \boldsymbol{\theta_t})} \qquad (9)$$

Then, it can be shown that the label posteriors are given as:

$$p(\boldsymbol{y_t}|\boldsymbol{w_t}, \mathbf{\Omega}) = \sum_{c_t=1}^{C} p(c_t|\boldsymbol{x_t}, \mathbf{\Omega}) p(\boldsymbol{y_t}|\boldsymbol{w_t}, c_t, \mathcal{W}_c) \qquad (10)$$

3. Experimental Setup

We applied the proposed model to the task of phone recognition on TIMIT database [7], which has been popular as a choice for reporting initial results for new methods of acoustic modeling, even in many recent DNN based methods. We use the KALDI speech recognition toolkit [9] for our experiments. In line with the standard TIMIT recipe in Kaldi, first the 462 speaker training set (with all SA records removed) is used to train a GMM-HMM triphone model over 48 phones using tied HMM states. Features used for this training are 13-dimensional MFCC feature vectors extracted over 25 ms frames with a shift of 10 ms, along with their first and second order derivatives. Then, features are spliced temporally, and the dimensionality of spliced frames is reduced to 40 using LDA. Another GMM-based triphone model is then trained using these features along with MLLT and SAT transforms. Further details about the training procedure can be obtained in [9]. The sequence of pdf-ids that represent tied HMM states is then obtained by forced alignment, to be used as labels $\boldsymbol{Y}$ in the model. A total of 1965 pdf-ids are used in the model. Once the alignments are obtained, the model training is carried out as follows:

1. **Initialization**: A diagonal GMM of C components is trained over all frames of training data to initialize the parameters $\boldsymbol{\theta_c}$ of the EGMLNN model. The Neural Network parameters $\mathcal{W}_c$ are initialized randomly.
2. **Component Posteriors**: Component posteriors are obtained using equation (6).
3. **Parameter Update**: Component priors and GMM parameters are updated as given by equations (8).

   For Neural Network training, first a 10% portion of the training set is held out for cross-validation. Neural Network weights for each component are updated through frame-weighted backpropagation over 90% of the training set using component posteriors as frame weights. An empirically chosen initial learning rate of 0.008 is used for the first epoch. This learning rate is continued as long as the cross-entropy loss function over cross-validation set keeps reducing. When the cross-entropy rises, the learning rate is halved for each successive epoch until the reduction in cross-entropy is negligible or negative.
4. **Iteration**: Further iterations of steps 2 and 3 are carried out until convergence.

From our observation, accuracy improvement generally stagnated after the first iteration of training, so only one iteration of training is used. A context window $\boldsymbol{w_t}$ of 11 frames is used as the input to Neural Networks. Each frame within $\boldsymbol{w_t}$ is mean and variance normalized using parameters $\boldsymbol{\theta_c}$ before supplying it as an input to the corresponding Neural Network.

During decoding, component posteriors are first obtained using equation (9). Only the top M posteriors are retained, and the rest are floored to zero in order to reduce the number of Neural Networks required for computation. Then, pdf-id posteriors are obtained using equation (10), using only M terms in the summation. In fact, we observed that there was a negligible degradation in accuracy between using $M = 1$ as against using $M = C$. Hence, all the results reported in the next section are evaluated using just $M = 1$. The language model used for decoding is a bigram model over phones, estimated from the training set. After decoding, the set of 48 phones is mapped to 39 phones as in [10] for scoring.

We compare our model to a baseline of using just one Neural Network for obtaining pdf-id posteriors. The procedure for initialization and training the baseline Neural Network is exactly the same as that used for training the Neural Networks in EGMLNN. The results on the 400 speaker development set are used to tune model parameters like the number of components, hidden layer size as well as depth of Neural Networks. The best models are then evaluated on the 192 speaker core test set.

4. Results

4.1. Comparision Metrics

4.1.1. Error Measures

We report two error measures for each model: The framewise error in recognizing true pdf-id, denoted by Frame Error Rate (FER), and the overall error in decoding the actual phone sequence, denoted as Phone Error Rate (PER).

4.1.2. Number of Computations

We compare the testing complexity of models, denoted by $\mathcal{T}$, in terms of the approximate number of arithmetic operations necessary per test frame to evaluate the label posteriors $p(\boldsymbol{y}|\boldsymbol{x})$.

The calculation of label posteriors for EGMLNN model requires two major operations: calculating $c^* = \arg\max_c p(c|\boldsymbol{x})$ followed by evaluation of $p(\boldsymbol{y}|\boldsymbol{x}, c^*)$. (As mentioned earlier in Section 3, we use just 1 Neural Network per test frame). If the input dimension is d_i, the output dimension is d_o, the hidden layer size is d_h, the Neural Network depth is N_h, and the number of components in the EGMLNN model is C, the approximate number of arithmetic operations required is as follows:

Gaussian Mixture Posteriors:

$$\arg\max_c p(c|\boldsymbol{x}) = \arg\min_c \sum_{i=1}^{d} \left(\frac{x_i - \mu_{ci}}{\sigma_{ci}}\right)^2 \qquad (11)$$

This requires approximately $4Cd_i$ operations (For each component: one subtraction, one division, and one squaring operation per dimension, and a sum across all dimensions).

Neural Network: One forward pass involves the following matrix multiplications (The computation involved in bias addition, sigmoid/softmax transformations is relatively much smaller and can be ignored):

- Input to first hidden layer: $d_i d_h$ operations
- first to last hidden layer: $(N_h - 1)d_h^2$ operations
- last hidden layer to output: $d_h d_o$ operations

In practice, $4Cd_i \ll d_i d_h + (N_h - 1)d_h^2 + d_o d_h$, therefore

$$\mathcal{T} \approx d_i d_h + (N_h - 1)d_h^2 + d_o d_h \qquad (12)$$

4.2. Dev set Results

The PER and FER on the dev set for the baseline model (single Neural Network) for different depths N_h and hidden layer sizes d_h are reported in Table 1. As expected, increasing the hidden layer size as well as depth generally leads to improved results.

Table 1: Baseline PER and FER (in %) on the dev set

N_h	FER			PER		
	d_h			d_h		
	500	*750*	*1000*	*500*	*750*	*1000*
1	50.4	49.8	49.3	20.3	19.7	19.7
2	49.0	48.5	48.0	19.7	19.4	19.1
3	48.9	48.1	47.5	19.3	18.8	18.9
4	49.0	48.1	47.7	19.1	18.5	18.6
5	48.9	48.5	47.8	18.8	19.0	18.4
6	49.6	48.8	48.3	19.3	18.9	18.5

The results obtained using EGMLNN model on the dev set for different values of C, d_h and N_h are summarized in Table 2. It can be seen that for a fixed C, increasing d_h or the N_h generally leads to better results. However, the trend with increasing C is interesting: for shallow networks of depth 1, the results generally improve as number of components is increased. In contrast, for deeper networks, the opposite trend is observed. This is possibly due to overfitting over the training set.

Comparing the two models, it can be seen that for the same value of d_h and N_h (i.e. the same test complexity $\mathcal{T}$), EGMLNN has lower FER and PER compared to the baseline, indicating the merit of using an ensemble as opposed to a single DNN of same complexity. The best performing models for both the baseline and EGMLNN have the same PER, but the EGMLNN model has a slightly lower FER. The best EGMLNN model also uses 3 hidden layers as opposed to 5 for the baseline, thereby reducing the test complexity by 30% (as given by equation (12)).

Table 2: EGMLNN PER and FER (in %) on the dev set

N_h	$C = 10$					
	FER			PER		
	d_h			d_h		
	500	*750*	*1000*	*500*	*750*	*1000*
1	49.2	48.7	48.5	19.5	19.3	19.1
2	48.1	47.7	47.5	19.0	18.8	18.8
3	48.0	47.3	47.0	18.8	18.8	18.4
	$C = 25$					
1	48.8	48.5	48.4	19.1	19.2	19.2
2	47.7	47.5	47.3	19.0	18.7	18.9
3	47.7	47.2	47.0	18.8	18.7	18.5
	$C = 50$					
1	48.8	48.3	48.4	19.0	19.0	18.9
2	48.0	47.7	47.6	18.7	18.6	18.6
3	47.7	47.4	47.3	18.9	18.7	18.6

4.3. Test set Results

It is clear from the dev set results that using $d_h = 1000$ consistently gives the best results for both the baseline and EGMLNN. Similarly, for the EGMLNN model, the best results are obtained by using $N_h = 3$. Having fixed these parameter values, we compare the three best baseline models ($N_h = 3, 4, 5$) to the three best EGMLNN models ($C = 10, 25, 50$) on the test set. The results are summarized in Table 3.

Table 3: Results on the test set

Baseline			EGMLNN		
N_h	FER	PER	C	FER	PER
4	48.1	20.1	**10**	47.4	19.4
5	48.5	20.0	**25**	47.4	19.5
6	48.9	19.8	**50**	47.8	19.7

It can be seen that the best EGMLNN model is able to obtain lower PER and FER compared to the best baseline model. In addition, it also uses 3 hidden layers as opposed to 6 for the baseline, reducing the test complexity $\mathcal{T}$ by 40%.

5. Conclusions and Future Work

We have presented Ensemble of Gaussian Mixture Localized Neural Networks, a model for jointly estimating the density of input variables, as well that of output variables conditioned on the input, for any mapping. We have shown that this approach can be viewed as a generalization of GMM and DNN based approaches for acoustic modeling. Major benefits of using this approach include the parallelizability of the training process, as well as reduced test complexity. In addition, the model is also able to obtain a lower Phone Error Rate compared to a a single DNN baseline as reported in results on the TIMIT database.

As future work, the results need to be verified by experimentation on larger datasets. On the model architecture front, the effect of using different classifiers such as CNNs [11] can be studied. The model also allows for many different initialization strategies which have not been addressed in this initial effort. For example, local DNNs can be initialized either by posterior-weighted component-wise pre-training [12], or by a globally pre-trained DNN, or even by a globally trained DNN. Therefore, our focus in future work will be on establishing best practices for model architecture, initialization and training.

6. References

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 160–167.

[4] L. Deng and J. C. Platt, "Ensemble deep learning for speech recognition," in *Proceedings of the Annual Conference of International Speech Communication Association (INTERSPEECH)*, 2014.

[5] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

[6] L. Xu, M. I. Jordan, and G. E. Hinton, "An alternative model for mixtures of experts," *Advances in Neural Information Processing Systems 7*, pp. 633–640, 1995.

[7] "TIMIT Acoustic-Phonetic Continuous Speech Corpus," https://catalog.ldc.upenn.edu/LDC93S1.

[8] A.-r. Mohamed, T. N. Sainath, G. Dahl, B. Ramabhadran, G. E. Hinton, and M. A. Picheny, "Deep belief networks using discriminative features for phone recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5060–5063.

[9] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," 2011.

[10] K.-F. Lee and H.-W. Hon, "Speaker-independent phone recognition using hidden markov models," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 37, no. 11, pp. 1641–1648, 1989.

[11] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4277–4280.

[12] G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.