Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition

George E. Dahl, Dong Yu, *Senior Member, IEEE*, Li Deng, *Fellow, IEEE*, and Alex Acero, *Fellow, IEEE*

***Abstract*—We propose a novel context-dependent (CD) model for large-vocabulary speech recognition (LVSR) that leverages recent advances in using deep belief networks for phone recognition. We describe a pre-trained deep neural network hidden Markov model (DNN-HMM) hybrid architecture that trains the DNN to produce a distribution over senones (tied triphone states) as its output. The deep belief network pre-training algorithm is a robust and often helpful way to initialize deep neural networks generatively that can aid in optimization and reduce generalization error. We illustrate the key components of our model, describe the procedure for applying CD-DNN-HMMs to LVSR, and analyze the effects of various modeling choices on performance. Experiments on a challenging business search dataset demonstrate that CD-DNN-HMMs can significantly outperform the conventional context-dependent Gaussian mixture model (GMM)-HMMs, with an absolute sentence accuracy improvement of 5.8% and 9.2% (or relative error reduction of 16.0% and 23.2%) over the CD-GMM-HMMs trained using the minimum phone error rate (MPE) and maximum-likelihood (ML) criteria, respectively.**

***Index Terms*—Artificial neural network–hidden Markov model (ANN-HMM), context-dependent phone, deep belief network, deep neural network hidden Markov model (DNN-HMM), speech recognition, large-vocabulary speech recognition (LVSR).**

I. Introduction

EVEN after decades of research and many successfully deployed commercial products, the performance of automatic speech recognition (ASR) systems in real usage scenarios lags behind human level performance (e.g., [2], [3]). There have been some notable recent advances in discriminative training (see an overview in [4]; e.g., maximum mutual information (MMI) estimation [5], minimum classification error (MCE) training [6], [7], and minimum phone error (MPE) training [8], [9]), in large-margin techniques (such as large-margin estimation [10], [11], large-margin hidden Markov model (HMM) [12], large-margin MCE [13]–[16], and boosted MMI [17]), as well as in novel acoustic models (such as conditional random fields (CRFs) [18]–[20], hidden CRFs [21], [22], and segmental CRFs [23]). Despite these advances, the elusive goal of human level accuracy in real-world conditions requires continued, vibrant research.

Recently, a major advance has been made in training densely connected, directed belief nets with many hidden layers. The resulting deep belief nets learn a hierarchy of nonlinear feature detectors that can capture complex statistical patterns in data. The deep belief net training algorithm suggested in [24] first initializes the weights of each layer individually in a purely unsupervised[1] way and then fine-tunes the entire network using labeled data. This semi-supervised approach using deep models has proved effective in a number of applications, including coding and classification for speech, audio, text, and image data ([25]–[29]). These advances triggered interest in developing acoustic models based on pre-trained neural networks and other deep learning techniques for ASR. For example, context-independent pre-trained, deep neural network HMM hybrid architectures have recently been proposed for phone recognition [30]–[32] and have achieved very competitive performance. Using pre-training to initialize the weights of a deep neural network has two main potential benefits that have been discussed in the literature. In [33], evidence was presented that is consistent with viewing pre-training as a peculiar sort of data-dependent regularizer whose effect on generalization error does not diminish with more data, even when the dataset is so vast that training cases are never repeated. The regularization effect from using information in the distribution of inputs can allow highly expressive models to be trained on comparably small quantities of labeled data. Additionally, [34], [33], and others have also reported experimental evidence consistent with pre-training aiding the subsequent optimization, typically performed by stochastic gradient descent. Thus, pre-trained neural networks often also achieve lower training error than neural networks that are not pre-trained (although this effect can often be confounded by the use of early stopping). These effects are especially pronounced in deep autoencoders.

Deep belief network pre-training was the first pre-training method to be widely studied, although many other techniques now exist in the literature (e.g., [35]). After [34] showed that deep auto-encoders could be trained effectively using deep belief net pre-training, there was a resurgence of interest in using deeper neural networks for applications. Although less pathological deep architectures than deep autoencoders can in some cases be trained without pre-training, for many problems and model architectures, researchers have reported pre-training to be

Manuscript received September 08, 2010; revised January 04, 2011, March 13, 2011; accepted March 14, 2011. Date of publication April 05, 2011; date of current version December 16, 2011. G. E. Dahl contributed to this work as an intern with Microsoft Research, Redmond, WA. This manuscript greatly extends the work presented at ICASSP 2011 [1]. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Nelson Morgan.

G. E. Dahl is with the Department of Computer Science, University of Toronto, Toronto, ON M5A 2N4, Canada (e-mail: gdahl@cs.toronto.edu).

D. Yu, L. Deng, and A. Acero are with the Speech Research Group, Microsoft Research, Redmond, WA 98034 USA (e-mail: dongyu@microsoft.com; deng@microsoft.com; alexac@microsoft.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASL.2011.2134090

[1]In the context of ASR, we use the term "unsupervised" to mean acoustic data with no transcriptions of any kind.

helpful (even in some cases for large single hidden layer neural networks trained on massive datasets, as in [28]). We view the various unsupervised pre-training techniques as convenient and robust ways to help train neural networks with many hidden layers that are generally helpful, rarely hurtful, and sometimes essential.

In this paper, we propose a novel acoustic model, a hybrid between a pre-trained, deep neural network (DNN) and a context-dependent (CD) hidden Markov model. The pre-training algorithm we use is the deep belief network (DBN) pre-training algorithm of [24], but we will denote our model with the abbreviation DNN-HMM to help distinguish it from a dynamic Bayes net (which we will not abreviate in this article) and to make it clear that we abandon the deep belief network once pre-training is complete and only retain and continue training the recognition weights. CD-DNN-HMMs combine the representational power of deep neural networks and the sequential modeling ability of context-dependent hidden Markov models (HMMs). In this paper, we illustrate the key ingredients of the model, describe the procedure to learn the CD-DNN-HMMs' parameters, analyze how various important design choices affect the recognition performance, and demonstrate that CD-DNN-HMMs can significantly outperform strong discriminatively-trained context-dependent Gaussian mixture model hidden Markov model (CD-GMM-HMM) baselines on the challenging business search dataset of [36], collected under actual usage conditions. To our best knowledge, this is the first time DNN-HMMs, which are formerly only used for phone recognition, are successfully applied to large-vocabulary speech recognition (LVSR) problems.

A. Previous Work Using Neural Network Acoustic Models

The combination of artificial neural networks (ANNs) and HMMs as an alternative paradigm for ASR started between the end of 1980s and the beginning of the 1990s. A variety of different architectures and training algorithms have been proposed in the literature (see the comprehensive survey in [37]). Among these techniques, the ones most relevant to this work are those that use the ANNs to estimate the HMM state-posterior probabilities [38]–[45], which have been referred to as ANN-HMM hybrid models in the literature. In these ANN-HMM hybrid architectures, each output unit of the ANN is trained to estimate the posterior probability of a continuous density HMMs' state given the acoustic observations. ANN-HMM hybrid models were seen as a promising technique for LVSR in the mid-1990s. In addition to their inherently discriminative nature, ANN-HMMs have two additional advantages: the training can be performed using the embedded Viterbi algorithm and the decoding is generally quite efficient.

Most early work (e.g., [39] and [38]) on the hybrid approach used context-independent phone states as labels for ANN training and considered small vocabulary tasks. ANN-HMMs were later extended to model context-dependent phones and were applied to mid-vocabulary and some large-vocabulary ASR tasks (e.g., in [45], which also employed recurrent neural architectures). However, in earlier work on context dependent ANN-HMM hybrid architectures [46], the posterior probability of the context-dependent phone was modeled as either

$$p(s_i, c_j \mid \mathbf{x}_t) = p(s_i \mid x_t) p(c_i \mid s_j, \mathbf{x}_t) \quad (1)$$

or

$$p(s_i, c_j \mid \mathbf{x}_t) = p(c_i \mid x_t) p(s_i \mid c_j, \mathbf{x}_t) \quad (2)$$

where $\mathbf{x}_t$ is the acoustic observation at time t, c_j is one of the clustered context classes $C = \{c_1, \ldots, c_J\}$, s_i is either a context-independent phone or a state in a context-independent phone. ANNs were used to estimate $p(s_i \mid \mathbf{x}_t)$ and $p(c_i \mid s_j, \mathbf{x}_t)$ (alternatively $p(c_i \mid \mathbf{x}_t)$ and $p(s_i \mid c_j, \mathbf{x}_t)$). Note that although these types of context-dependent ANN-HMMs outperformed GMM-HMMs for some tasks, the improvements were small.

These earlier hybrid attempts had some important limitations. For example, using only backpropagation to train the ANN makes it challenging (although not impossible) to exploit more than two hidden layers well and the context-dependent model described above does not take advantage of the numerous effective techniques developed for GMM-HMMs. Around 1999, the desire to use HMM advances from the speech research community directly without developing replacement techniques and tools contributed to a shift from using neural nets to predict phonetic states to using neural nets to augment features for later use in a conventional GMM-HMM recognizer (e.g., [47]). In this work, however, we do not take that approach, but instead we try to improve the earlier hybrid approaches by replacing more traditional neural nets with deeper, pre-trained neural nets and by using the senones [48] (tied triphone states) of a GMM-HMM tri-phone model as the output units of the neural network, in line with state-of-the-art HMM systems.

Although this work uses the hybrid approach, as alluded to above, much recent work using neural networks in acoustic modeling uses the so-called TANDEM approach, first proposed in [49]. The TANDEM approach augments the input to a GMM-HMM system with features derived from the suitably transformed output of one or more neural networks, typically trained to produce distributions over monophone targets. In a similar vein, [50] uses features derived from an earlier "bottle-neck" hidden layer instead of using the neural network outputs directly. Many recent papers (e.g., [51]–[54]) train neural networks on LVSR datasets (often in excess of 1000 hours of data) and use variants of these approaches, either augmenting the input to a GMM-HMM system with features based on the neural network outputs or some earlier hidden layer. Although a neural network nominally containing three hidden layers (the largest number of layers investigated in [55]) might be used to create bottle-neck features, if the feature layer is the middle hidden layer then the resulting features are only produced by an encoder with a single hidden layer.

Neural networks for producing bottle-neck features are very similar architecturally to autoencoders since both typically have a small code layer. Deeper neural networks, especially deeper autoencoders, are known to be difficult to train with backpropagation alone. For example, [34] reports in one experiment that they are unable to get results nearly so good as those possible with deep belief network pre-training when training a deep (the encoder and decoder in their architecture both had three hidden

layers) autoencoder with a nonlinear conjugate gradient algorithm. Both [56] and [57] investigate why training deep feed-forward neural networks can often be easier with some form of pre-training or a sophisticated optimizer of the sort used in [58].

Since the time of the early hybrid architectures, the vector processing capabilities of modern GPUs and the advent of more effective training algorithms for deep neural nets have made much more powerful architectures feasible. Much previous hybrid ANN-HMM work focused on context-independent or rudimentary context-dependent phone models and small to mid-vocabulary tasks (with notable exceptions such as [45]), possibly masking some of the potential advantages of the ANN-HMM hybrid approach. Additionally, GMM-HMM training is much easier to parallelize in a computer cluster setting, which historically gave such systems a significant advantage in scalability. Also, since speaker and environment adaptation is generally easier for GMM-HMM systems, the GMM-HMM approach has been the dominant one in the past two decades for speech recognition. That being said, if we consider the wider use of neural networks in acoustic modeling beyond the hybrid approach, neural network feature extraction is an important component of many state-of-the-art acoustic models.

B. Introduction to the DNN-HMM Approach

The primary contributions of this work are the development of a context-dependent, pre-trained, deep neural network HMM hybrid acoustic model (CD-DNN-HMM); a description of our recipe for applying this sort of model to LVSR problems; and an analysis of our results which show substantial improvements in recognition accuracy for a difficult LVSR task over discriminatively-trained pure CD-GMM-HMM systems. Our work differs from earlier context-dependent ANN-HMMs [42], [41] in two key respects. First, we used deeper, more expressive neural network architectures and thus employed the unsupervised DBN pre-training algorithm to make sure training would be effective. Second, we used posterior probabilities of senones (tied triphone HMM states) [48] as the output of the neural network, instead of the combination of context-independent phone and context class used previously in hybrid architectures. This second difference also distinguishes our work from earlier uses of DNN-HMM hybrids for phone recognition [30]–[32], [59]. Note that [59], which also appears in this issue, is the context-independent version of our approach and builds the foundation for our work. The work in this paper focuses on context-dependent DNN-HMMs using posterior probabilities of senones as network outputs and can be successfully applied to large vocabulary tasks. Training the neural network to predict a distribution over senones causes more bits of information to be present in the neural network training labels. It also incorporates context-dependence into the neural network outputs (which, since we are not using a Tandem approach, lets us use a decoder based on triphone HMMs), and it may have additional benefits. Our evaluation was done on LVSR instead of phoneme recognition tasks as was the case in [30]–[32], [59]. It represents the first large-vocabulary application of a pre-trained, deep neural network approach. Our results show that our CD-DNN-HMM system provides dramatic improvements over a discriminatively trained CD-GMM-HMM baseline.

The remainder of this paper is organized as follows. In Section II, we briefly introduce RBMs and deep belief nets, and outline the general pre-training strategy we use. In Section III, we describe the basic ideas, the key properties, and the training and decoding strategies of our CD-DNN-HMMs. In Section IV, we analyze experimental results on a 65 K+ vocabulary business search dataset collected from the Bing mobile voice search application (formerly known as Live Search for mobile [36], [60]) under real usage scenarios. Section V offers conclusions and directions for future work.

II. Deep Belief Networks

Deep belief networks (DBNs) are probabilistic generative models with multiple layers of stochastic hidden units above a single bottom layer of observed variables that represent a data vector. DBNs have undirected connections between the top two layers and directed connections to all other layers from the layer above. There is an efficient unsupervised algorithm, first described in [24], for learning the connection weights in a DBN that is equivalent to training each adjacent pair of layers as an restricted Boltzmann machine (RBM). There is also a fast, approximate, bottom-up inference algorithm to infer the states of all hidden units conditioned on a data vector. After the unsupervised *pre-training* phase, Hinton *et al.* [24] used the *up-down* algorithm to optimize all of the DBN weights jointly. During this *fine-tuning* phase, a supervised objective function could also be optimized.

In this paper, we use the DBN weights resulting from the unsupervised pre-training algorithm to initialize the weights of a deep, but otherwise standard, feed-forward neural network and then simply use the backpropagation algorithm [61] to fine-tune the network weights with respect to a supervised criterion. Pre-training followed by stochastic gradient descent is our method of choice for training deep neural networks because it often outperforms random initialization for the deeper architectures we are interested in training and provides results very robust to the initial random seed. The generative model learned during pre-training helps prevent overfitting, even when using models with very high capacity and can aid in the subsequent optimization of the recognition weights.

Although empirical results ultimately are the best reason for the use of a technique, our motivation for even trying to find and apply deeper models that might be capable of learning rich, distributed representations of their input is also based on formal and informal arguments by other researchers in the machine learning community. As argued in [62] and [63], insufficiently deep architectures can require an exponential blow-up in the number of computational elements needed to represent certain functions satisfactorily. Thus, one primary motivation for using deeper models such as neural networks with many layers is that they have the potential to be much more representationally efficient for some problems than shallower models like GMMs. Furthermore, GMMs as used in speech recognition typically have a large number of Gaussians with independently parameterized means which may result in those Gaussians being highly localized and thus may result in such models only performing local generalization. In effect, such a GMM would partition the input space into regions each modeled by a single Gaussian.

[64] proved that constant leaf decision trees require a number of training cases exponential in their input dimensionality to learn certain rapidly varying functions. [64] also makes more general and less formal arguments that models that create a single hard or soft partitioning of the input space and use separately parameterized simple models for each region are doomed to have similar generalization issues when trained on rapidly varying functions. In a related vein, [65] also proves an analogous "curse of rapidly-varying functions" for a large class of local kernel machines that include both supervised learning algorithms (e.g., SVMs with Gaussian kernels) and many semi-supervised algorithms and unsupervised manifold learning algorithms. It is our fear that functions important for solving difficult perceptual tasks in domains such as computer vision and computer audition will have a componential structure that makes them vary rapidly even though there is perhaps only a comparatively small number of factors that cause these variations. Although it remains to be seen to what extent these arguments about architectural depth and local generalization apply to speech recognition, one of our hopes in this work is to demonstrate that replacing GMMs with deeper models can reduce recognition error in a difficult LVSR task, even if we are unable to show that our proposed system performs well because of some sort of avoidance of the potential issues we discuss above.

A. Restricted Boltzmann Machines

RBMs [66] are a type of undirected graphical model constructed from a layer of binary stochastic hidden units and a layer of stochastic visible units that, for the purposes of this work, will either be Bernoulli or Gaussian distributed conditional on the hidden units. The visible and hidden units form a bipartite graph with no visible-visible or hidden-hidden connections. For concreteness, we will assume the visible units are binary for the moment (we always assume binary hidden units in this work) and describe how we deal with real-valued speech data at the end of this section. An RBM assigns an energy to every configuration of visible and hidden state vectors, denoted $\mathbf{v}$ and $\mathbf{h}$ respectively, according to

$$E(\mathbf{v},\mathbf{h}) = -\mathbf{b}^{\mathrm{T}}\mathbf{v} - \mathbf{c}^{\mathrm{T}}\mathbf{h} - \mathbf{v}^{\mathrm{T}}\mathbf{W}\mathbf{h} \tag{3}$$

where $\mathbf{W}$ is the matrix of visible/hidden connection weights, $\mathbf{b}$ is a visible unit bias, and $\mathbf{c}$ is a hidden unit bias. The probability of any particular setting of the visible and hidden units is given in terms of the energy of that configuration by

$$P(\mathbf{v},\mathbf{h}) = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{Z} \tag{4}$$

where the normalization factor $Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$ is known as the partition function.

The lack of direct connections within each layer enables us to derive simple exact expressions for $P(\mathbf{v}\,|\,\mathbf{h})$ and $P(\mathbf{h}\,|\,\mathbf{v})$, since the visible units are conditionally independent given the hidden unit states and vice versa. We perform this derivation for $P(\mathbf{h}\,|\,\mathbf{v})$ below. We will refer to the term in (3) dependent on h_i as $\gamma_i(\mathbf{v},h_i) = -(c_i + \mathbf{v}^{\mathrm{T}}\mathbf{W}_{*,i})h_i$, with $\mathbf{W}_{*,i}$ denoting the ith column of $\mathbf{W}$. Starting with the definition of $P(\mathbf{h}\,|\,\mathbf{v})$, we obtain (see [62] for another version of this derivation along with other useful ones)

$$\begin{aligned}
P(\mathbf{h}\,|\,\mathbf{v}) &= \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\tilde{\mathbf{h}}} e^{-E(\mathbf{v},\tilde{\mathbf{h}})}} \\
&= \frac{e^{\mathbf{b}^{\mathrm{T}}\mathbf{v}+\mathbf{c}^{\mathrm{T}}\mathbf{h}+\mathbf{v}^{\mathrm{T}}\mathbf{W}\mathbf{h}}}{\sum_{\tilde{\mathbf{h}}} e^{\mathbf{b}^{\mathrm{T}}\mathbf{v}+\mathbf{c}^{\mathrm{T}}\tilde{\mathbf{h}}+\mathbf{v}^{\mathrm{T}}\mathbf{W}\tilde{\mathbf{h}}}} \\
&= \frac{e^{\mathbf{c}^{\mathrm{T}}\mathbf{h}+\mathbf{v}^{\mathrm{T}}\mathbf{W}\mathbf{h}}}{\sum_{\tilde{\mathbf{h}}} e^{\mathbf{c}^{\mathrm{T}}\tilde{\mathbf{h}}+\mathbf{v}^{\mathrm{T}}\mathbf{W}\tilde{\mathbf{h}}}} \\
&= \frac{\prod_i e^{c_i h_i + \mathbf{v}^{\mathrm{T}}\mathbf{W}_{*,i}h_i}}{\sum_{\tilde{h}_1}\cdots\sum_{\tilde{h}_N}\prod_i e^{c_i\tilde{h}_i+\mathbf{v}^{\mathrm{T}}\mathbf{W}_{*,i}\tilde{h}_i}} \\
&= \frac{\prod_i e^{-\gamma_i(\mathbf{v},h_i)}}{\sum_{\tilde{h}_1}\cdots\sum_{\tilde{h}_N}\prod_i e^{-\gamma_i(\mathbf{v},\tilde{h}_i)}} \\
&= \frac{\prod_i e^{-\gamma_i(\mathbf{v},h_i)}}{\prod_i\sum_{\tilde{h}_i} e^{-\gamma_i(\mathbf{v},\tilde{h}_i)}} \\
&= \prod_i \frac{e^{-\gamma_i(\mathbf{v},h_i)}}{\sum_{\tilde{h}_i} e^{-\gamma_i(\mathbf{v},\tilde{h}_i)}} \qquad (5) \\
&= \prod_i P(h_i\,|\,\mathbf{v}). \qquad (6)
\end{aligned}$$

Since the $h_i \in \{0,1\}$, the sum in the denominator of (5) has only two terms and thus

$$\begin{aligned}
P(h_i = 1\,|\,\mathbf{v}) &= \frac{e^{-\gamma_i(\mathbf{v},1)}}{e^{-\gamma_i(\mathbf{v},1)} + e^{-\gamma_i(\mathbf{v},0)}} \\
&= \sigma(c_i + \mathbf{v}^{\mathrm{T}}\mathbf{W}_{*,i})
\end{aligned}$$

yielding

$$P(\mathbf{h} = \mathbf{1}\,|\,\mathbf{v}) = \sigma(\mathbf{c} + \mathbf{v}^{\mathrm{T}}\mathbf{W}) \tag{7}$$

where σ denotes the (elementwise) logistic sigmoid, $\sigma(x) = (1+e^{-x})^{-1}$. For the binary visible unit case to which we restrict ourselves to at the moment, a completely symmetric derivation lets us obtain

$$P(\mathbf{v} = \mathbf{1}\,|\,\mathbf{h}) = \sigma(\mathbf{b} + \mathbf{h}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}). \tag{8}$$

The form of (7) is what allows us to use the weights of an RBM to initialize a feed-forward neural network with sigmoidal hidden units because we can equate the inference for RBM hidden units with forward propagation in a neural network.

Before writing an expression for the log probability assigned by an RBM to some visible vector $\mathbf{v}$, it is convenient to define a quantity known as the free energy:

$$F(\mathbf{v}) = -\log\left(\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}\right).$$

Using $F(\mathbf{v})$, we can write the per-training-case log likelihood as

$$\ell(\boldsymbol{\theta}) = -F(\mathbf{v}) - \log\left(\sum_{\boldsymbol{\nu}} e^{-F(\boldsymbol{\nu})}\right)$$

with $\boldsymbol{\theta}$ denoting the model parameters.

To train an RBM, we perform stochastic gradient descent on the negative log likelihood. In the experiments in this work, we

use the following expression for the $(t+1)$th weight update for some typical model parameter w_{ij}:

$$\Delta w_{ij}(t+1) = m\Delta w_{ij}(t) - \alpha\frac{\partial \ell}{\partial w_{ij}} \tag{9}$$

where α is the learning rate/step size and m is the "momentum" factor used to smooth out the weight updates. Unlike in a GMM, in an RBM the gradient of the log likelihood of the data is not feasible to compute exactly. The general form of the derivative of the log likelihood of the data is

$$-\frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \left\langle \frac{\partial E}{\partial \boldsymbol{\theta}} \right\rangle_{\text{data}} - \left\langle \frac{\partial E}{\partial \boldsymbol{\theta}} \right\rangle_{\text{model}}.$$

In particular, for the visible-hidden weight updates we have

$$-\frac{\partial \ell(\boldsymbol{\theta})}{\partial w_{ij}} = \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}}. \tag{10}$$

The first expectation $\langle v_i h_j \rangle_{\text{data}}$ is the frequency with which the visible unit v_i and the hidden unit h_j are on together in the training set and $\langle v_i h_j \rangle_{\text{model}}$ is that same expectation under the distribution defined by the model. Unfortunately, the term $\langle \cdot \rangle_{\text{model}}$ takes exponential time to compute exactly, so we are forced to use an approximation. Since RBMs are in the intersection between Boltzmann machines and product of experts models, they can be trained using contrastive divergence as described in [67]. The one-step contrastive divergence approximation for the gradient with respect to the visible-hidden weights is

$$-\frac{\partial \ell(\boldsymbol{\theta})}{\partial w_{ij}} \approx \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_1 \tag{11}$$

where $\langle \cdot \rangle_1$ denotes the expectation over one-step reconstructions. In other words, an expectation computed with samples generated by running the Gibbs sampler [defined using (7) and (8)] initialized at the data for one full step. Similar update rules for the other model parameters are easy to derive by simply replacing $(\partial E)/(\partial w_{ij}) = v_i h_j$ in (11) with the appropriate partial derivative of the energy function (or by creating a hidden unit and a visible unit both with the constant activation of one to derive the updates for the biases).

Although RBMs with the energy function of (3) are suitable for binary data, in speech recognition the acoustic input is typically represented with real-valued feature vectors. The Gaussian–Bernoulli restricted Boltzmann machine (GRBM) only requires a slight modification of (3) (see [68] for a generalization of RBMs to any distribution in the exponential family). The GRBM energy function we use in this work is given by

$$E(\mathbf{v}, \mathbf{h}) = \frac{1}{2}(\mathbf{v} - \mathbf{b})^{\mathrm{T}}(\mathbf{v} - \mathbf{b}) - \mathbf{c}^{\mathrm{T}}\mathbf{h} - \mathbf{v}^{\mathrm{T}}\mathbf{W}\mathbf{h}. \tag{12}$$

Note that (12) implicitly assumes that the visible units have a diagonal covariance Gaussian noise model with a variance of 1 on each dimension. In the GRBM case, (7) does not change, but (8) becomes

$$P(\mathbf{v} \,|\, \mathbf{h}) = \mathcal{N}(\mathbf{v}; \mathbf{b} + \mathbf{h}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}, I)$$

where I is the appropriate identity matrix. However, when actually training a GRBM and creating a reconstruction, we never actually sample from the distribution above; we simply set the visible units to be equal to their means. The only difference between our training procedure for GRBMs using the energy function in (12) and binary RBMs using the energy function in (3) is how the reconstructions are generated, all positive and negative statistics used for gradients are the same.

B. Deep Belief Network Pre-Training

Now that we have described using contrastive divergence to train an RBM and the two types of RBMs we use in this work, we will discuss how to perform deep belief network pre-training. Once we have trained an RBM on data, we can use the RBM to re-represent our data. For each data vector $\mathbf{v}$, we use (7) to compute a vector of hidden unit activation probabilities $\mathbf{h}$. We use these hidden activation probabilities as training data for a new RBM. Thus each set of RBM weights can be used to extract features from the output of the previous layer. Once we stop training RBMs, we have the initial values for all the weights of the hidden layers of a neural net with a number of hidden layers equal to the number of RBMs we trained. With pre-training complete, we add a randomly initialized softmax output layer and use backpropagation to fine-tune all the weights in the network discriminatively. Since only the supervised fine-tuning phase requires labeled data, we can potentially leverage a large quantity of unlabeled data during pre-training, although this capability is not yet important for our LVSR experiments [69] due to the abundance of weakly supervised data.

III. CD-DNN-HMM

Hidden Markov models (HMMs) have been the dominant technique for LVSR for at least two decades. An HMM is a generative model in which the observable acoustic features are assumed to be generated from a hidden Markov process that transitions between states $S = \{s_1, \ldots, s_K\}$. The key parameters in the HMM are the initial state probability distribution $\pi = \{p(q_0 = s_i)\}$, where q_t is the state at time t; the transition probabilities $a_{ij} = p(q_t = s_j \,|\, q_{t-1} = s_i)$; and a model to estimate the observation probabilities $p(\mathbf{x}_t \,|\, s_i)$.

In conventional HMMs used for ASR, the observation probabilities are modeled using GMMs. These GMM-HMMs are typically trained to maximize the likelihood of generating the observed features. Recently, discriminative training strategies such as MMI [5], MCE [6], [7], MPE [8], [9], and large-margin techniques [10]–[17] have been proposed. The potential of these discriminative techniques, however, is restricted by the limitations of the GMM emission distribution model. The recently proposed CRF [18]–[20] and HCRF [21], [22] models use log-linear models to replace GMM-HMMs. These models typically use manually designed features and have been shown to be equivalent to the GMM-HMM [20] in their modeling ability if only the first and second order statistics are used as the features.

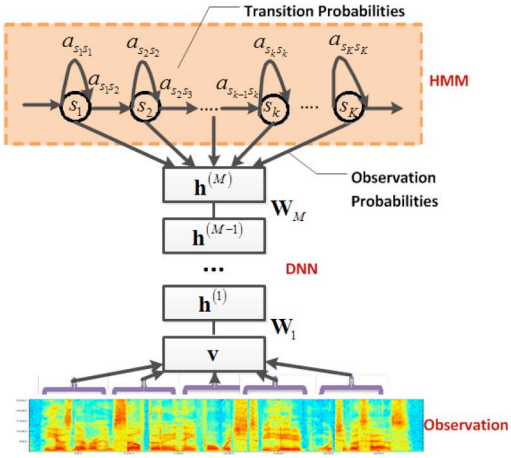


Fig. 1. Diagram of our hybrid architecture employing a deep neural network. The HMM models the sequential property of the speech signal, and the DNN models the scaled observation likelihood of all the senones (tied tri-phone states). The same DNN is replicated over different points in time.

A. Architecture of CD-DNN-HMMs

Fig. 1 illustrates the architecture of our proposed CD-DNN-HMMs. The foundation of the hybrid approach is the use of a forced alignment to obtain a frame level labeling for training the ANN. The key difference between the CD-DNN-HMM architecture and earlier ANN-HMM hybrid architectures (and context-independent DNN-HMMs) is that we model senones as the DNN output units directly. The idea of using senones as the modeling unit has been proposed in [22] where the posterior probabilities of senones were estimated using deep-structured conditional random fields (CRFs) and only one audio frame was used as the input of the posterior probability estimator. This change offers two primary advantages. First, we can implement a CD-DNN-HMM system with only minimal modifications to an existing CD-GMM-HMM system, as we will show in Section II-B. Second, any improvements in modeling units that are incorporated into the CD-GMM-HMM baseline system, such as cross-word triphone models, will be accessible to the DNN through the use of the shared training labels.

If DNNs can be trained to better predict senones, then CD-DNN-HMMs can achieve better recognition accuracy than tri-phone GMM-HMMs. More precisely, in our CD-DNN-HMMs, the decoded word sequence $\hat{w}$ is determined as

$$\hat{w} = \arg\max_{w} p(w \mid \mathbf{x}) = \arg\max_{w} p(\mathbf{x} \mid w) p(w) / p(\mathbf{x}) \quad (13)$$

where $p(w)$ is the language model (LM) probability, and

$$p(\mathbf{x} \mid w) = \sum_{q} p(\mathbf{x}, q \mid w) p(q \mid w) \quad (14)$$

$$\cong \max_{\pi}(q_0) \prod_{t=1}^{T} a_{q_{t-1} q_t} \prod_{t=0}^{T} p(\mathbf{x}_t \mid q_t) \quad (15)$$

is the acoustic model (AM) probability. Note that the observation probability is

$$p(\mathbf{x}_t \mid q_t) = p(q_t \mid \mathbf{x}_t) p(\mathbf{x}_t) / p(q_t) \quad (16)$$

where $p(q_t \mid \mathbf{x}_t)$ is the state (senone) posterior probability estimated from the DNN, $p(q_t)$ is the prior probability of each state (senone) estimated from the training set, and $p(\mathbf{x}_t)$ is independent of the word sequence and thus can be ignored. Although dividing by the prior probability $p(q_t)$ (called scaled likelihood estimation by [38], [40], [41]) may not give improved recognition accuracy under some conditions, we have found it to be very important in alleviating the label bias problem, especially when the training utterances contain long silence segments.

B. Training Procedure of CD-DNN-HMMs

CD-DNN-HMMs can be trained using the embedded Viterbi algorithm. The main steps involved are summarized in Algorithm 1, which takes advantage of the triphone tying structures and the HMMs of the CD-GMM-HMM system. Note that the logical triphone HMMs that are effectively equivalent are clustered and represented by a physical triphone (i.e., several logical triphones are mapped to the same physical triphone). Each physical triphone has several (typically 3) states which are tied and represented by senones. Each senone is given a *senoneid* as the label to fine-tune the DNN. The *state2id* mapping maps each physical triphone state to the corresponding *senoneid*.

Algorithmic 1 Main Steps to Train CD-DNN-HMMs

1) Train a best tied-state CD-GMM-HMM system where state tying is determined based on the data-driven decision tree. Denote the CD-GMM-HMM *gmm-hmm*.
2) Parse *gmm-hmm* and give each senone name an ordered *senoneid* starting from 0. The *senoneid* will be served as the training label for DNN fine-tuning.
3) Parse *gmm-hmm* and generate a mapping from each physical tri-phone state (e.g., b-ah+t.s2) to the corresponding *senoneid*. Denote this mapping *state2id*.
4) Convert *gmm-hmm* to the corresponding CD-DNN-HMM *dnn–hmm*1 by borrowing the tri-phone and senone structure as well as the transition probabilities from *gmm–hmm*.
5) Pre-train each layer in the DNN bottom-up layer by layer and call the result *ptdnn*.
6) Use *gmm–hmm* to generate a state-level alignment on the training set. Denote the alignment *align–raw*.
7) Convert *align–raw* to *align* where each physical tri-phone state is converted to *senoneid*.
8) Use the *senoneid* associated with each frame in *align* to fine-tune the DBN using back-propagation or other approaches, starting from *ptdnn*. Denote the DBN *dnn*.
9) Estimate the prior probability $p(s_i) = n(s_i)/n$, where $n(s_i)$ is the number of frames associated with senone s_i in *align* and n is the total number of frames.
10) Re-estimate the transition probabilities using *dnn* and *dnn–hmm*1 to maximize the likelihood of observing the features. Denote the new CD-DNN-HMM *dnn–hmm*2.
11) Exit if no recognition accuracy improvement is observed in the development set; Otherwise use

dnn and $dnn–hmm2$ to generate a new state-level alignment $align–raw$ on the training set and go to Step 7.

To support the training and decoding of CD-DNN-HMMs, we needed to develop a series of tools, the most important of which were: 1) the tool to convert the CD-GMM-HMMs to CD-DNN-HMMs; 2) the tool to do forced alignment using CD-DNN-HMMs; and 3) the CD-DNN-HMM decoder. We have found that it is relatively easy to develop these tools by modifying the corresponding HTK tools if the format of the CD-DNN-HMM model files is wisely specified.

In our specific implementation, each senone in the CD-DNN-HMM is identified as a (pseudo) single Gaussian whose dimension equals the total number of senones. The variance (precision) of the Gaussian is irrelevant, so it can be set to any positive value (e.g., always set to 1). The value of the first dimension of each senone's mean is set to the corresponding $senoneid$ determined in Step 2 in Algorithm 1. The values of other dimensions are not important and can be set to any value such as 0. Using this trick, evaluating each senone is equivalent to a table lookup of the features (log-likelihood) produced by the DNN with the index indicated by the $senoneid$.

IV. Experimental Results

To evaluate the proposed CD-DNN-HMMs and to understand the effect of different decisions made at each step of CD-DNN-HMM training, we have conducted a series of experiments on a business search dataset collected from the Bing mobile voice search application (formerly known as Live Search for mobile [36], [60])—a real-world large-vocabulary spontaneous speech recognition task. In this section, we report our experimental setup and results, demonstrate the efficacy of the proposed approach, and analyze the training and decoding time.

A. Dataset Description

The Bing mobile voice search application allows users to do US-wide business and web search from their mobile phones via voice. The business search dataset used in our experiments was collected under real usage scenarios in 2008, at which time the application was restricted to do location and business lookup. All audio files collected were sampled at 8 kHz and encoded with the GSM codec. Some examples of typical queries in the dataset are "Mc-Donalds," "Denny's restaurant," and "oak ridge church." This is a challenging task since the dataset contains all kinds of variations: noise, music, side-speech, accents, sloppy pronunciation, hesitation, repetition, interruption, and different audio channels.

The dataset was split into a training set, a development set, and a test set. To simulate the real data collection and training procedure, and to avoid having overlap between training, development, and test sets, the dataset was split based on the time stamp of the queries. All queries in the training set were collected before those in the development set, which were in turn collected before those in the test set. For the sake of easy comparisons, we have used the public lexicon from Carnegie Mellon University. The normalized nationwide language model (LM) used in the evaluation contains 65 K word unigrams, 3.2 million word bi-grams, and 1.5 million word tri-grams, and was trained using the data feed and collected query logs; the perplexity is 117.

TABLE I
Information on the Business Search Dataset

	Hours	Number of Utterances
Training Set	24	32,057
Development Set	6.5	8,777
Test Set	9.5	12,758

Table I summarizes the number of utterances and total duration of audio files (in hours) in the training, development, and test sets. All 24 hours of training data included in the training set are manually transcribed. We used 24 hours of training data in this study since it lets us run more experiments (training our CD-DNN-HMM systems is time consuming compared to training CD-GMM-HMMs).

Performance on this task was evaluated using sentence accuracy (SA) instead of word accuracy for a variety of reasons. In order to compare our results with [70], we would need to compute sentence accuracy anyway. The average sentence length is 2.1 tokens, so sentences are typically quite short. Also, the users care most about whether they can find the business or location they seek in the fewest attempts. They typically will repeat what they have said if one of the words is mis-recognized. Additionally, there is significant inconsistency in spelling that makes using sentence accuracy more convenient. For example, "Mc-Donalds" sometimes is spelled as "McDonalds," "Walmart" sometimes is spelled as "Wal-mart," and "7-eleven" sometimes is spelled as "7 eleven" or "seven-eleven." For these reasons, when calculating sentence accuracy we concatenate all the words in the utterance and remove hyphens and apostrophes before comparing the recognition outputs with the references so that we can remove some of the effects caused by the LM and poor text normalization and focus on the AM. The sentence out-of-vocabulary rate (OOV) using the 65 K vocabulary LM is 6% on both the development and test sets. In other words, the best possible SA we can achieve is 94% using this setup.

B. CD-GMM-HMM Baseline Systems

To compare our proposed CD-DNN-HMM model with standard discriminatively trained, GMM-based systems, we have trained clustered cross-word triphone GMM-HMMs with maximum-likelihood (ML), maximum mutual information (MMI), and minimum phone error (MPE) criteria. The 39-dim features used in the experiments include the 13-dim static Mel-frequency cepstral coefficient (MFCC) (with C0 replaced with energy) and its first and second derivatives. The features were pre-processed with the cepstral mean normalization (CMN) algorithm.

We optimized the baseline systems by tuning the tying structures, number of senones, and Gaussian splitting strategies on the development set. The performance of the best CD-GMM-HMM configuration is summarized in Table II. All systems reported in II have 53 K logical and 2 K physical tri-phones with 761 shared states (senones), each of which is a GMM with 24 mixture components. Note that our ML

TABLE II
CD-GMM-HMM BASELINE RESULTS

Criterion	Dev Accuracy	Test Accuracy
ML	62.9%	60.4%
MMI	65.1%	62.8%
MPE	65.5%	63.8%

TABLE III
PERFORMANCE OF SINGLE HIDDEN LAYER MODELS USING MONOPHONE AND TRIPHONE HMM ALIGNMENT LABELS

Alignment	# Hidden Units	Label	Dev Accuracy
Monophone	1.5K	Monophone State	55.5%
Triphone	1.5K	Monophone State	59.1%

baseline of 60.4% trained using 24 hours of data is only 2.5% worse than the 62.9% obtained in [70], even though the latter used 130 hours of manually transcribed data and about 2000 hours of user-click confirmed data (90% accuracy). This small difference in accuracy indicates that the baseline we compare with in this paper is not weak. Since we did not personally obtain the result from [70], there may be other differences between our setup and the one used in [70] in addition to the larger training set.

The discriminative training of the CD-GMM-HMM was carried out using the HTK.[2] The lattices were generated using HDecode[3] and, when generating the lattices, the weak word unigram LM estimated from the training transcription was used. As shown in Table II, the MPE-trained CD-GMM-HMM outperformed both the ML- and MMI-trained CD-GMM-HMM with a sentence accuracy of 65.5% and 63.8% on the development and test sets respectively.

C. CD-DNN-HMM Results and Analysis

Many decisions need to be made when training CD-DNN-HMMs. In this subsection, we will examine how these choices affect recognition accuracy. In particular, we will empirically compare the performance difference between using a monophone alignment and a tri-phone alignment, using monophone state labels and tri-phone senone labels, using 1.5 K and 2 K hidden units in each layer, using an ANN-HMM and a DNN-HMM, and tuning and not tuning the transition probabilities. For all experiments reported below, we have used 11 frames (5-1-5) of MFCCs as the input features of the DNNs, following [30] and [31]. During pre-training we used a learning rate of 0.004 for all layers. For fine-tuning, we used a learning rate of 0.08 for the first 6 epochs and a learning rate of 0.002 for the last 6 epochs. In all our experiments, we averaged updates over minibatchs of 256 training cases before applying them. To all weight updates, we added a "momentum" term of 0.9 times the previous update [see (9)]. We selected the values of these hyperparameters by

[2]The lattice probability scale factor `LATPROBSCALE` was set to 1/`LMW` where `LMW` is the LM weight, i-smooth parameters `ISMOOTHTAU`, `ISMOOTHTAUT`, and `ISMOOTHTAUW` were set to 100, 10, and 10, respectively, for the MMI training, and 50, 10, and 10, respectively, for the MPE training.

[3]We used HDecode.exe with command line parameters "-t 250.0 -v 200.0 -u 5000 -n 32 -s 15.0 -p 0.0" for the denominator and "-t 1500.0 -n 64 -s 15.0 -p 0.0" for the numerator.

TABLE IV
COMPARISON OF CONTEXT-INDEPENDENT MONOPHONE STATE LABELS AND CONTEXT-DEPENDENT TRIPHONE SENONE LABELS

# Hidden Layers	# Hidden Units	Label Type	Dev Accuracy
1	2K	Monophone States	59.3%
1	2K	Triphone Senones	68.1%
3	2K	Monophone States	64.2%
3	2K	Triphone Senones	69.6%

TABLE V
CONTEXT-DEPENDENT MODELS WITH AND WITHOUT PRE-TRAINING

Model Type	# Hidden Layers	# Hidden Units	Dev Accuracy
without pre-training	1	2K	68.0%
without pre-training	2	2K	68.2%
with pre-training	1	2K	68.1%
with pre-training	2	2K	69.5%

hand, based on preliminary single hidden layer experiments so it may be possible to obtain even better performance with the deeper models using a more exhaustive hyperparameter search.

Our first experiment used an alignment generated from a monophone GMM-HMM and used the monophone states as the DNN training labels. Such a setup only achieved 55.5% sentence accuracy on the development set if a single 1.5 K hidden layer is used, as shown in Table III. Switching to an alignment generated from an ML-trained triphone GMM-HMM, but still using monophone states as labels for the DNN, increased accuracy to 59.1%.

The performance can be further improved to 59.3% if we use 2 K instead of 1.5 K hidden units, as shown in Table IV. However, an even larger performance improvement occurred when we used triphone senones as the DNN training labels, which yields 68.1% sentence accuracy on the development set, even with only one hidden layer. Note that this accuracy is already 2.6% higher than the 65.5% achieved using the MPE-trained CD-GMM-HMMs. The accuracy increased to 69.6% when three hidden layers were used. Table IV shows that models trained using senone labels perform much better than those trained using monophone state labels when either one or three hidden layers were used. Using senone labels has been the single largest source of improvement of all the design decisions we analyzed.

An obvious question to ask is whether the pre-training step in the DNN is truly necessary or helpful. To answer this question, we compared CD-DNN-HMMs with and without pre-training in Table V. As expected, if only one hidden layer was used, systems with and without pre-training have comparable performance. However, when two hidden layers were used, the accuracy of 69.6% obtained with pre-training applied noticeably surpassed the accuracy of 68.2% obtained without pre-training on the development set. The pre-trained two layer model had a frame-level misclassification rate of 31.13%, whereas the un-pre-trained two layer model had a frame-level misclassification rate of 32.83%. The cross entropy loss per case of the two hidden layer models was 1.73 and 1.18

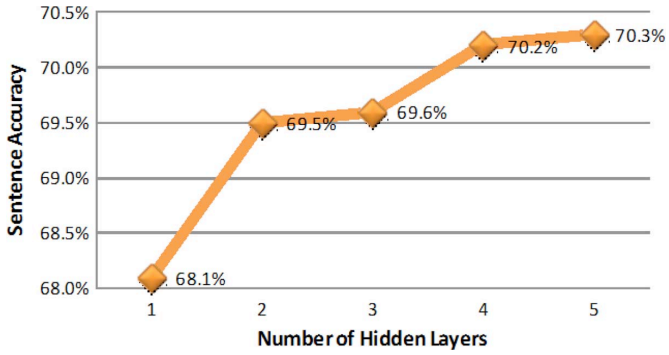


Fig. 2. Relationship between the recognition accuracy and the number of layers. Context-dependent models with 2 K hidden units per layer were used to obtain the results.

bits, respectively. Our general anecdotal experience (built in part from other speech datasets) has been that pre-training on acoustic data never hurts the frame-level error of models we try and can be especially helpful when using very large models. Even the largest models we use in this work are comparable in size to ones used on TIMIT by [30], even though we use a much larger dataset here. We hope to use much larger models still in the future and make better use of the regularization effect of generative pre-training. That being said, the pre-training phase seems to give a clear improvement in the two hidden layer experiment we describe in Table V.

Fig. 2 demonstrates how the sentence accuracy improves as more layers are added in the CD-DNN-HMM. When three hidden layers were used, the accuracy increased to 69.6%. The accuracy further improved to 70.2% with four hidden layers and 70.3% with five hidden layers. Overall, using the five hidden-layer models provides us with a 2.2% accuracy improvement over the single hidden-layer system when the same alignment is used. Although it is possible that using even more than five hidden layers would continue to improve the accuracy, we expect any such gains to be modest at best, so we restricted ourselves to at most five hidden layers in the rest of this work.

In order to demonstrate the efficiency of parameterization enjoyed by deeper neural networks, we have also trained a single hidden layer neural network with 16 K hidden units, a number chosen to guarantee that the weights required a little more space to store than the weights for our five hidden layer models. We were able to obtain an accuracy of 68.6% on the development set, which is slightly more than the 2K hidden unit single layer result of 68.1% in Fig. 2, but well below even the two layer result of 69.5% (let alone the five layer result of 70.3%).

Table VI shows our results after the main steps of Algorithm 1. All systems in Table VI use a DNN with five hidden layers of 2K units each and senone labels. As we have shown in Table III, using a better alignment to generate training labels for the DNN can improve the accuracy. This observation is also confirmed in Table VI. Using alignments generated with MPE-trained CD-GMM-HMMs, we can obtain 70.7% and 68.8% accuracies on the development and test sets, respectively. These results are 0.4% higher than those we achieved using the ML CD-GMM-HMM alignments.

TABLE VI
EFFECTS OF ALIGNMENT AND TRANSITION PROBABILITY TUNING ON BEST DNN ARCHITECTURE

Alignment	Tune Trans.	Dev Acc	Test Acc
from CD-GMM-HMM ML	no	70.3%	68.4%
from CD-GMM-HMM MPE	no	70.7%	68.8%
from CD-GMM-HMM MPE	yes	71.0%	69.0%
from CD-DNN-HMM	no	71.7%	69.6%
from CD-DNN-HMM	yes	71.8%	69.6%

TABLE VII
SUMMARY OF TRAINING TIME USING 24 HOURS OF TRAINING DATA AND 2 K HIDDEN UNITS PER LAYER

Type	# of Layers	Time Per Epoch	# of Epochs
Pre-train	1	0.2 h	50
Pre-train	2	0.5 h	20
Pre-train	3	0.6 h	20
Pre-train	4	0.7 h	20
Pre-train	5	0.8 h	20
Fine-tune	4	1.2 h	12
Fine-tune	5	1.4 h	12

Table VI also demonstrates that tuning the transition probabilities in the CD-DNN-HMMs also seems to help slightly. Tuning the transition probabilities comes with another benefit. When we use transition probabilities directly borrowed from the CD-GMM-HMMs, the best decoding performance usually was obtained when the AM weight was set to 2. However, after tuning the transition probabilities, we no longer need to tune the AM weights.

Once we have trained our best CD-DNN-HMM using a CD-GMM-HMM alignment, we can use the CD-DNN-HMM to generate an even better alignment. Table VI shows that the accuracies on the development and test sets can be increased to 71.7% and 69.6%, respectively, from 71.0% and 69.0%, which were obtained using dnn–$hmm1$. Tuning the transition probabilities again only marginally improves the performance. Overall, our proposed CD-DNN-HMMs obtained 69.6% accuracy on the test set, which is 5.8% (or 9.2%) higher than those obtained using the MPE (or ML)-trained CD-GMM-HMMs. This improvement translates to a 16.0% (or 23.2%) relative error rate reduction over the MPE (or ML)-trained CD-GMM-HMMs and is statistically significant at significant level of 1% according to McNemar's test.

D. Training and Decoding Time

We have just shown that CD-DNN-HMMs substantially outperform CD-GMM-HMMs in terms of recognition accuracy on our task. A natural question to ask is whether the gain was obtained at a significantly higher computational cost for training and decoding.

Table VII summarizes the DNN training time using 24 hours of training data, 2 K hidden units, and 11 frames of MFCCs as input features. The time recorded in the table is based on a trainer written in Python. The training was carried out on a Dell Precision T3500 workstation, which is a quad core computer with a CPU clock speed of 2.66 GHz, 8 MB of L3 CPU cache,

TABLE VIII
SUMMARY OF DECODING TIME

Processing Unit	# of Layers	DNN Time Per Frame	Search Time Per Frame	Real-time Factor
CPU	4	4.3 ms	1.5 ms	0.58
GPU	4	0.16 ms	1.5 ms	0.17
CPU	5	5.2 ms	1.5 ms	0.67
GPU	5	0.20 ms	1.5 ms	0.17

and 12 GB of 1066 MHz DDR3 SDRAM. The training also used an NVIDIA Tesla C1060 general purpose graphical processing unit (GPGPU), which contains 4 GB of GDDR3 RAM and 240 processing cores. We used the CUDAMat library [71] to perform matrix operations on the GPU from our Python code.

From Table VII we can observe that to train a five-layer CD-DNN-HMM, pre-training takes about $0.2 \times 50 + 0.5 \times 20 + 0.6 \times 20 + 0.7 \times 20 + 0.8 \times 20 = 62$ hours. Fine-tuning takes about $1.4 \times 12 = 16.8$ hours. To achieve the best result reported in this paper, we have to run two passes of fine-tuning, one with the MPE CD-GMM-HMM alignment, and one with the CD-DNN-HMM alignment. The total fine-tuning time is thus $16.8 \times 2 = 33.6$ hours. To train the system, we also need to spend time to normalize the MFCC features to allow each to have zero-mean and unit-variance, and to generate alignments. However, these tasks can be easily parallelized and the time spent on them is very small compared to the DNN training time. The total time spent to train the system from scratch is about four days. We have observed that using a GPU speeds up training by about a factor of 30 faster than just using the CPU in our setup. Without using a GPU, it would take about three months to train the best system.

The bottleneck in the training process is the mini-batch stochastic gradient descend (SGD) algorithm used to train the DNNs. SGD is inherently sequential and is difficult to parallelize across machines. So far SGD with a GPU is the best training strategy for CD-DNN-HMMs since the GPU at least can exploit the parallelism in the layered DNN structure.

When more training data is available, the time spent on each epoch increases. However, fewer epochs will be needed when more training data is available. We speculate that using a strategy similar to our current one described in this paper, it should be possible to train an effective CD-DNN-HMM system that exploits 2000 hours of training data in about 50 days (using a single GPU).

While training is considerably more expensive than for CD-GMM-HMM systems, decoding is still very efficient. Table VIII summarizes the decoding time on our four and five-layer 2K hidden unit CD-DNN-HMM systems with and without using GPUs. Note that in our implementation, the search is always done using CPUs. It takes only 0.58 and 0.67 times real time to decode with four and five-layer CD-DNN-HMMs, respectively, without using GPUs. Using a GPU reduces decoding time to 0.17 times real time, at which point DNN computations no longer dominate. For reference, our baseline CD-GMM-HMM system decodes in 0.54 times real time.

V. CONCLUSION AND FUTURE WORK

We have described a context-dependent DNN-HMM model for LVSR that achieves substantially better results than strong, discriminatively trained CD-GMM-HMM baselines on a challenging business search dataset. Although our experiments show that CD-DNN-HMMs provide dramatic improvements in recognition accuracy, training CD-DNN-HMMs is quite expensive compared to training CD-GMM-HMMs (although on a similar scale as other neural-network-based acoustic models and certainly feasible for large datasets, if one can afford weeks of training time). This is primarily because the CD-DNN-HMM training algorithms we have discussed are not easy to parallelize across computers and need to be carried out on a single GPU machine. That being said, decoding in CD-DNN-HMMs is very efficient so test time is not an issue in real-world applications.

We believe our work on CD-DNN-HMMs is only the first step towards a more powerful acoustic model for LVSR; many issues remain to be resolved. Here are a few we view as particularly important. First, although CD-DNN-HMM training is asymptotically quite scalable, in practice it is quite challenging to train CD-DNN-HMMs on tens of thousands of hours of data. To achieve this level of practical scalability, we must parallelize training not just at the matrix arithmetic level. Finding new ways to parallelize training may require a better theoretical understanding of deep learning. Second, we must find highly effective speaker and environment adaptation algorithms for DNN-HMMs, ideally ones that are completely unsupervised and integrated with the pre-training phase. Inspiration for such algorithms may come from the ANN-HMM literature (e.g., [72] and [73]) or the many successful adaptation techniques developed in the past decades for GMM-HMMs (e.g., MLLR [74], MAP [75], joint compensation of distortions [76], variable parameter HMMs [77]). Third, the training in this study used the embedded Viterbi algorithm, which is not optimal. We believe additional improvement may be achieved by optimizing an objective function based on the full sequence, as we have already demonstrated on the TIMIT dataset with some success [31]. In addition, we view the treatment of the time dimension of speech by DNN-HMM and GMM-HMMs alike as a very crude way of dealing with the intricate temporal properties of speech. The weaknesses in how HMMs deal with the temporal dimension of speech inputs have been analyzed in detail in [78]–[81]. There is a vast space to explore in the deep learning framework using the insights gained from temporal-centric generative modeling research in neural networks and in speech (e.g., [2], [47], [82], and [83]). Finally, although Gaussian RBMs can learn an initial distributed representation of their input, they still produce a diagonal covariance Gaussian for the conditional distribution over the input space given the latent state (as diagonal covariance GMMs also do). A more powerful first layer model, namely the mean-covariance restricted Boltzmann machine [84] significantly enhanced the performance of context-independent DNN-HMMs for phone recognition in [32]. We therefore view applying similar models to LVSR as an enticing area of future work.

ACKNOWLEDGMENT

The authors would like to thank Dr. P. Nguyen at Microsoft Research for preparing the dataset, providing the ML-trained CD-GMM-HMM baseline, and engaging in valuable discussions. They would also like to thank Dr. C. Liu of the Microsoft Corporation speech product team for his assistance in getting the discriminatively-trained CD-GMM-HMM baselines, Dr. J. Droppo at Microsoft Research for his parallel computing support, Prof. G. Hinton at the University of Toronto for advice and encouragement of this work, and Prof. N. Morgan at the University of California-Berkeley for discussions on prior work on ANN-HMM approaches.

REFERENCES

[1] G. Dahl, D. Yu, L. Deng, and A. Acero, "Large vocabulary continuous speech recognition with context-dependent DBN-HMMs," in *Proc. ICASSP*, 2011.

[2] J. M. Baker, L. Deng, J. Glass, S. Khudanpur, C. Lee, N. Morgan, and D. O'Shaugnessy, "Research developments and directions in speech recognition and understanding, part 1," *IEEE Signal Process. Mag.*, vol. 26, no. 3, pp. 75–80, May 2009.

[3] J. M. Baker, L. Deng, J. Glass, S. Khudanpur, C. Lee, N. Morgan, and D. O'Shaugnessy, "Research developments and directions in speech recognition and understanding, part 2," *IEEE Signal Process. Mag.*, vol. 26, no. 4, pp. 78–85, Jul. 2009.

[4] X. He, L. Deng, and W. Chou, "Discriminative learning in sequential pattern recognition—A unifying review for optimization-oriented speech recognition," *IEEE Signal Process. Mag.*, vol. 25, no. 5, pp. 14–36, Sep. 2008.

[5] S. Kapadia, V. Valtchev, and S. J. Young, "MMI training for continuous phoneme recognition on the TIMIT database," in *Proc. ICASSP*, 1993, vol. 2, pp. 491–494.

[6] B. H. Juang, W. Chou, and C. H. Lee, "Minimum classification error rate methods for speech recognition," *IEEE Trans. Speech Audio Process.*, vol. 5, no. 3, pp. 257–265, May 1997.

[7] E. McDermott, T. Hazen, J. L. Roux, A. Nakamura, and S. Katagiri, "Discriminative training for large vocabulary speech recognition using minimum classification error," *IEEE Trans. Speech Audio Process.*, vol. 15, no. 1, pp. 203–223, Jan. 2007.

[8] D. Povey and P. Woodland, "Minimum phone error and i-smoothing for improved discriminative training," in *Proc. ICASSP*, 2002, pp. 105–108.

[9] D. Povey, "Discriminative training for large vocabulary speech recognition," Ph.D. dissertation, Eng. Dept., Cambridge Univ., Cambridge, U.K., 2003.

[10] X. Li, H. Jiang, and C. Liu, "Large margin HMMs for speech recognition," in *Proc. ICASSP*, 2005, pp. 513–516.

[11] H. Jiang and X. Li, "Incorporating training errors for large margin HMMs under semi-definite programming framework," in *Proc. ICASSP*, 2007, vol. 4, pp. 629–632.

[12] F. Sha and L. Saul, "Large margin Gaussian mixture modeling for phonetic classification and recognition," in *Proc. ICASSP*, 2006, pp. 265–268.

[13] D. Yu, L. Deng, X. He, and A. Acero, "Use of incrementally regulated discriminative margins in MCE training for speech recognition," in *Proc. ICSLP*, 2006, pp. 2418–2421.

[14] D. Yu and L. Deng, "Large-margin discriminative training of hidden Markov models for speech recognition," in *Proc. ICSC*, 2007, pp. 429–436.

[15] D. Yu, L. Deng, X. He, and A. Acero, "Large-margin minimum classification error training for large-scale speech recognition tasks," in *Proc. ICASSP*, 2007, vol. 4, pp. 1137–1140.

[16] D. Yu, L. Deng, X. He, and A. Acero, "Large-margin minimum classification error training a theoretical risk minimization perspective," *Comput. Speech Lang.*, vol. 22, no. 4, pp. 415–429, 2008.

[17] D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah, "Boosted MMI for model and feature space discriminative training," in *Proc. ICASSP*, 2008, pp. 4057–4060.

[18] Y. Hifny and S. Renals, "Speech recognition using augmented conditional random fields," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 17, no. 2, pp. 354–365, Feb. 2009.

[19] J. Morris and E. Fosler-Lussier, "Combining phonetic attributes using conditional random fields," in *Proc. Interspeech*, 2006, pp. 597–600.

[20] G. Heigold, "A Log-linear discriminative modeling framework for speech recognition," Ph.D. dissertation, Aachen Univ., Aachen, Germany, 2010.

[21] A. Gunawardana, M. Mahajan, A. Acero, and J. C. Platt, "Hidden conditional random fields for phone classification," in *Proc. Interspeech*, 2005, pp. 1117–1120.

[22] D. Yu and L. Deng, "Deep-structured hidden conditional random fields for phonetic recognition," in *Proc. Interspeech*, 2010, pp. 2986–2989.

[23] G. Zweig and P. Nguyen, "A segmental conditional random field toolkit for speech recognition," in *Proc. Interspeech*, 2010.

[24] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, pp. 1527–1554, 2006.

[25] V. Nair and G. Hinton, "3-D object recognition with deep belief nets," *Adv. Neural Inf. Process. Syst. 22*, pp. 1339–1347, 2009.

[26] R. Salakhutdinov and G. Hinton, "Semantic hashing," in *Proc. SIGIR Workshop Inf. Retrieval Applicat. Graph. Models*, 2007.

[27] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. 25th Int. Conf. Mach. Learn. ser. ICML'08*, New York, 2008, pp. 160–167, ACM.

[28] V. Mnih and G. Hinton, "Learning to detect roads in high-resolution aerial images," in *Proc. 11th Eur. Conf. Comput. Vision (ECCV)*, Sep. 2010.

[29] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?," in *Proc. Int. Conf. Comput. Vis. (ICCV'09). IEEE*, 2009.

[30] A. Mohamed, G. E. Dahl, and G. E. Hinton, "Deep belief networks for phone recognition," in *Proc. NIPS Workshop Deep Learn. Speech Recogn. Rel. Applicat.*, 2009.

[31] A. Mohamed, D. Yu, and L. Deng, "Investigation of full-sequence training of deep belief networks for speech recognition," in *Proc. Interspeech*, 2010, pp. 2846–2849.

[32] G. E. Dahl, M. Ranzato, A. Mohamed, and G. E. Hinton, J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds., "Phone recognition with the mean-covariance restricted Boltzmann machine," *Adv. Neural Inf. Process. Syst. 23*, pp. 469–477, 2010.

[33] D. Erhan, A. Courville, Y. Bengio, and P. Vincent, "Why does unsupervised pre-training help deep learning?," in *Proc. AISTATS'10*, May 2010, vol. 9, pp. 201–208.

[34] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[35] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, W. W. Cohen, A. McCallum, and S. T. Roweis, Eds., "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn. (ICML'08)*, 2008, pp. 1096–1103.

[36] A. Acero, N. Bernstein, R. Chambers, Y. Ju, X. Li, J. Odell, P. Nguyen, O. Scholtz, and G. Zweig, "Live search for mobile: Web services by voice on the cellphone," in *Proc. ICASSP*, 2008, pp. 5256–5259.

[37] E. Trentin and M. Gori, "A survey of hybrid ANN/HMM models for automatic speech recognition," *Neurocomputing*, vol. 37, pp. 91–126, 2001.

[38] H. Boulard and N. Morgan, "Continuous speech recognition by connectionist statistical methods," *IEEE Trans. Neural Netw.*, vol. 4, no. 6, pp. 893–909, Nov. 1993.

[39] N. Morgan and H. Bourlard, "Continuous speech recognition using multilayer perceptrons with hidden Markov models," in *Proc. ICASSP*, 1990, pp. 413–416.

[40] H. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*, ser. The Kluwer International Series in Engineering and Computer Science. Boston, MA: Kluwer, 1994, vol. 247.

[41] S. Renals, N. Morgan, H. Boulard, M. Cohen, and H. Franco, "Connectionist probability estimators in HMM speech recognition," *IEEE Trans. Speech Audio Process.*, vol. 2, no. 1, pp. 161–174, Jan. 1994.

[42] H. Franco, M. Cohen, N. Morgan, D. Rumelhart, and V. Abrash, "Context-dependent connectionist probability estimation in a hybrid hidden Markov model-neural net speech recognition system," *Comput. Speech Lang.*, vol. 8, pp. 211–222, 1994.

[43] J. Hennebert, C. Ris, H. Bourlard, S. Renals, and N. Morgan, "Estimation of global posteriors and forward-backward training of hybrid HMM/ANN systems," in *Proc. Eurospeech*, 1997, vol. 4, pp. 1951–1954.

[44] Y. Yan, M. Fanty, and R. Cole, "Speech recognition using neural networks with forward-backward probability generated targets," in *Proc. ICASSP*, 1997, pp. 3241–3244.

[45] A. J. Robinson, G. D. Cook, D. P. W. Ellis, E. Fosler-Lussier, S. J. Renals, and D. A. G. Williams, "Connectionist speech recognition of broadcast news," *Speech Commun.*, vol. 37, pp. 27–45, May 2002.

[46] H. Bourlard, N. Morgan, C. Wooters, and S. Renals, "CDNN: A context-dependent neural network for continuous speech recognition," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process. (ICASSP)*, San Francisco, CA, 1992, pp. 349–352.

[47] N. Morgan, Q. Zhu, A. Stolcke, K. Sonmez, S. Sivadas, T. Shinozaki, M. Ostendorf, P. Jain, H. Hermansky, D. Ellis, G. Doddington, B. Chen, O. Cetin, H. Bourlard, and M. Athineos, "Pushing the envelope—Aside," *IEEE Signal Process. Mag.*, vol. 22, no. 5, pp. 81–88, Sep. 2005.

[48] M. Hwang and X. Huang, "Shared-distribution hidden Markov models for speech recognition," *IEEE Trans. Speech Audio Process.*, vol. 1, no. 4, pp. 414–420, Jan. 1993.

[49] H. Hermansky, D. P. W. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional HMM systems," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2000, vol. 3, pp. 1635–1638.

[50] F. Grézl, M. Karafiát, S. Kontár, and J. Černocký, IEEE Signal Processing Society, "Probabilistic and bottle-neck features for LVCSR of meetings," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP'07)*, 2007, pp. 757–760.

[51] F. Valente, M. M. Doss, C. Plahl, S. Ravuri, and W. Wang, "A comparative large scale study of MLP features for mandarin ASR," in *Proc. Interspeech'10*, Brisbane, Australia, Sep. 2010, pp. 2630–2633.

[52] Q. Zhu, A. Stolcke, B. Y. Chen, and N. Morgan, "Using MLP features in SRI's conversational speech recognition system," in *Proc. Interspeech*, 2005, pp. 2141–2144.

[53] P. Fousek, L. Lamel, and J.-L. Gauvain, "Transcribing broadcast data using MLP features," in *Proc. Interspeech*, 2008, pp. 1433–1436.

[54] D. Vergyri, A. Mandal, W. Wang, A. Stolcke, J. Zheng, M. Graciarena, D. Rybach, C. Gollan, R. Schlüter, K. Kirchhoff, A. Faria, and N. Morgan, "Development of the SRI/Nightingale Arabic ASR system," in *Proc. Interspeech*, Brisbane, Australia, Sep. 2008, pp. 1437–1440.

[55] F. Gŕezl and P. Fousek, "Optimizing bottle-neck features for LVCSR," in *Proc. ICASSP. IEEE Signal Process. Soc.*, 2008, pp. 4729–4732.

[56] Y. Bengio and X. Glorot, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. AISTATS'10*, May 2010, vol. 9, pp. 249–256.

[57] D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent, "The difficulty of training deep architectures and the effect of unsupervised pre-training," in *Proc. 12h Int. Conf. Artif. Intell. Statist. (AISTATS'09)*, 2009, pp. 153–160.

[58] J. Martens, J. Fürnkranz and T. Joachims, Eds., "Deep learning via hessian-free optimization," in *Proc. 27th Int. Conf. Mach. Learn. (ICML-10)*, Haifa, Israel, Jun. 2010, pp. 735–742, Omnipress.

[59] A. Mohamed, G. E. Dahl, and G. E. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 19, no. 6, pp. XXXX–XXXX, Jul. 2011.

[60] D. Yu, Y. C. Ju, Y. Y. Wang, G. Zweig, and A. Acero, "Automated directory assistance system—From theory to practice," in *Proc. Interspeech*, 2007, pp. 2709–2711.

[61] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[62] Y. Bengio, "Learning deep architectures for AI," *Foundat. and Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[63] Y. Bengio and Y. LeCun, , L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, Eds., "Scaling learning algorithms towards AI," in *Large-Scale Kernel Machines*. Cambridge, MA: MIT Press, 2007.

[64] Y. Bengio, O. Delalleau, and C. Simard, "Decision trees do not generalize to new variations," *Comput. Intell.*, vol. 26, no. 4, pp. 449–467, Nov. 2010.

[65] Y. Bengio, O. Delalleau, and N. Le Roux, , Y. Weiss, B. Schölkopf, and J. Platt, Eds., "The curse of highly variable functions for local kernel machines," in *Advances in Neural Information Processing Systems 18*. Cambridge, MA: MIT Press, 2006, pp. 107–114.

[66] P. Smolensky, "Information processing in dynamical systems: Foundations of harmony theory," *Parall. Distrib. Process.*, vol. 1, pp. 194–281, 1986.

[67] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, pp. 1771–1800, 2002.

[68] M. Welling, M. Rosen-Zvi, and G. E. Hinton, "Exponential family harmoniums with an application to information retrieval," *Adv. Neural Inf. Process. Syst. 17*, 2004.

[69] D. Yu, L. Deng, and G. Dahl, "Roles of pre-training and fine-tuning in context-dependent DBN-HMMs for real-world speech recognition," in *Proc. NIPS 2010 Workshop Deep Learn. Unsupervised Feature Learn.*, 2010.

[70] G. Zweig and P. Nguyen, "A segmental CRF approach to large vocabulary continuous speech recognition," in *Proc. ASRU*, 2009, pp. 152–155.

[71] V. Mnih, "Cudamat: A CUDA-based matrix class for Python," Dept. of Comput. Sci., Univ. of Toronto, Tech. Rep. UTML TR 2009-004, Nov. 2009.

[72] J. Neto, L. Almeida, M. Hochberg, C. Martins, L. Nunes, S. Renals, and T. Robinson, "Speaker adaptation for hybrid HMM-ANN continuous speech recognition system," in *Proc. Eurospeech*, 1995, pp. 2171–2174.

[73] J. P. Neto, C. Martins, and L. B. Almeida, "Speaker-adaptation in a hybrid HMM-MLP recognizer," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1996, pp. 3382–3385.

[74] M. J. F. Gales and P. C. Woodland, "Mean and variance adaptation within the MLLR framework," *Comput. Speech Lang.*, vol. 10, pp. 249–264, 1996.

[75] C. Lee and Q. Huo, "On adaptive decision rules and decision parameter adaptation for automatic speech recognition," in *Proc. IEEE*, 2000, vol. 88, no. 8, pp. 1241–1269.

[76] J. Li, L. Deng, D. Yu, Y. Gong, and A. Acero, "A unified framework of HMM adaptation with joint compensation of additive and convolutive distortions," *Comput. Speech Lang.*, vol. 23, pp. 389–405, 2009.

[77] D. Yu, L. Deng, Y. Gong, and A. Acero, "A novel framework and training algorithm for variable-parameter hidden Markov models," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 17, no. 7, pp. 1348–1360, Sep. 2009.

[78] J. Bridle, L. Deng, J. Picone, H. Richards, J. Ma, T. Kamm, M. Schuster, S. Pike, and R. Regan, "An investigation of segmental hidden dynamic models of speech coarticulation for automatic speech recognition," in *Report 1998 Workshop Lang. Eng.*, 1998, pp. 1–61 [Online]. Available: http://www.clsp.jhu.edu/ws98/projects/dynamic/presentations/finalhtml, The Johns Hopkins University.

[79] L. Deng, "A dynamic, feature-based approach to the interface between phonology and phonetics for speech modeling and recognition," *Speech Commun.*, vol. 24, no. 4, pp. 299–323, 1998.

[80] L. Deng, D. Yu, and A. Acero, "Structured speech modeling," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, no. 5, pp. 1492–1504, Sep. 2006.

[81] L. Deng, D. Yu, and A. Acero, "A bidirectional target filtering model of speech coarticulation: Two-stage implementation for phonetic recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, no. 1, pp. 256–265, Jan. 2006.

[82] I. Sutskever, G. Hinton, and G. Taylor, "The recurrent temporal restricted Boltzmann machine," in *Proc. NIPS*, 2009.

[83] L. Deng, "Computational models for speech production," in *Computational Models of Speech Pattern Processing, (NATO ASI Series)*. New York: Springer, 1999, pp. 199–213.

[84] M. Ranzato and G. Hinton, "Modeling pixel means and covariances using factorized third-order Boltzmann machines," in *Proc. Comput. Vis. Pattern Recogn. Conf. (CVPR 2010)*, 2010, pp. 2551–2558.

George E. Dahl received the B.A. in computer science (highest honors) from Swarthmore College, Swarthmore, PA, and the M.Sc. degree from the University of Toronto, Toronto, ON, Canada, where he is currently pursuing the Ph.D. degree with a research focus in statistical machine learning.

His current main research interest is in training models that learn many levels of rich, distributed representations from large quantities of perceptual and linguistic data.

Dong Yu (M'97–SM'06) received the B.S. degree (with honors) in electrical engineering from Zhejiang University, Hangzhou, China, the M.S. degree in electrical engineering from the Chinese Academy of Sciences, Beijing, the M.S. degree in computer science from Indiana University at Bloomington, and the Ph.D. degree in computer science from University of Idaho, Moscow.

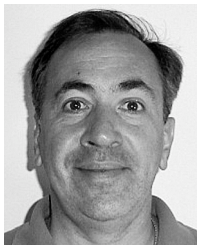
He joined Microsoft Corporation in 1998 and Microsoft Speech Research Group in 2002, where he is a Researcher. His current research interests include speech processing, robust speech recognition, discriminative training, spoken dialog system, voice search technology, machine learning, and pattern recognition. He has published more than 70 papers in these areas and is the inventor/coinventor of more than 40 granted/pending patents.

Dr. Dong Yu is a member of ACM and ISCA. He is currently serving as an associate editor of the *IEEE Signal Processing Magazine* and was the Lead Guest Editor of the IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING—Special Issue on Deep Learning for Speech and Language Processing. He is also serving as a Guest Professor at the University of Science and Technology of China.

Li Deng (S'85–M'86–SM'91–F'05) received the B.S. degree from the University of Science and Technology of China, Hefei, and the Ph.D. degree from the University of Wisconsin-Madison.

He joined the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, in 1989 as an Assistant Professor where he became a Full Professor with tenure in 1996. From 1992 to 1993, he conducted sabbatical research at the Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, and from 1997 to 1998, at ATR Interpreting Telecommunications Research Laboratories, Kyoto, Japan. In 1999, he joined Microsoft Research, Redmond, WA as a Senior Researcher, where he is currently a Principal Researcher. Since 2000, he has also been an Affiliate Professor in the Department of Electrical Engineering, University of Washington, Seattle, teaching the graduate course of Computer Speech Processing. His current (and past) research activities include automatic speech and speaker recognition, spoken language identification and understanding, speech-to-speech translation, machine translation, language modeling, statistical methods and machine learning, neural information processing, deep-structured learning, machine intelligence, audio and acoustic signal processing, statistical signal processing and digital communication, human speech production and perception, acoustic phonetics, auditory speech processing, auditory physiology and modeling, noise robust speech processing, speech synthesis and enhancement, multimedia signal processing, and multimodal human–computer interaction. In these areas, he has published over 300 refereed papers in leading journals and conferences, three books, 15 book chapters, and has given keynotes, tutorials, and lectures worldwide. He has been granted over 40 U.S. or international patents in acoustics/audio, speech/language technology, and other fields of signal processing.

Dr. Deng is a Fellow of the Acoustical Society of America. He received awards/honors bestowed by IEEE, ISCA, ASA, Microsoft, and other organizations. He serves on the Board of Governors of the IEEE Signal Processing Society (2008–2010), and as Editor-in-Chief for the *IEEE Signal Processing Magazine* (SPM, 2009–2011), which ranks consistently among the top journals with the highest citation impact. He is elected by the International Speech Communication Association (ISCA) as its Distinguished Lecturer in 2010–2011.



Alex Acero (S'85–M'90–SM'00–F'04) received the M.S. degree from the Polytechnic University of Madrid, Madrid, Spain, in 1985, the M.S. degree from Rice University, Houston, TX, in 1987, and the Ph.D. degree from Carnegie Mellon University, Pittsburgh, PA, in 1990, all in electrical engineering.

He worked in Apple Computer's Advanced Technology Group in 1990–1991. In 1992, he joined Telefonica I+D, Madrid, Spain, as Manager of the Speech Technology Group. Since 1994, he has been with Microsoft Research, Redmond, WA, as a Research Area Manager, directing an organization with over 50 engineers working on audio, multimedia, communication, speech, and natural language. He is also an affiliate Professor of Electrical Engineering at the University of Washington. He is an author of the books *Acoustical and Environmental Robustness in Automatic Speech Recognition* (Kluwer, 1993) and *Spoken Language Processing* (Prentice-Hall, 2001), has written invited chapters in four edited books and 200 technical papers. He holds 93 U.S. patents.

Dr. Acero has served the IEEE Signal Processing Society as Vice President Technical Directions (2007–2009), 2006 Distinguished Lecturer, member of the Board of Governors (2004–2005), Associate Editor for the IEEE SIGNAL PROCESSING LETTERS (2003–2005) and the IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING (2005–2007), and member of the editorial board of the IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING (2006–2008) and *IEEE Signal Processing Magazine* (2008–2010). He also served as member (1996–2000) and Chair (2000–2002) of the Speech Technical Committee of the IEEE Signal Processing Society. He was Publications Chair of ICASSP'98, Sponsorship Chair of the 1999 IEEE Workshop on Automatic Speech Recognition and Understanding, and General Co-Chair of the 2001 IEEE Workshop on Automatic Speech Recognition and Understanding.