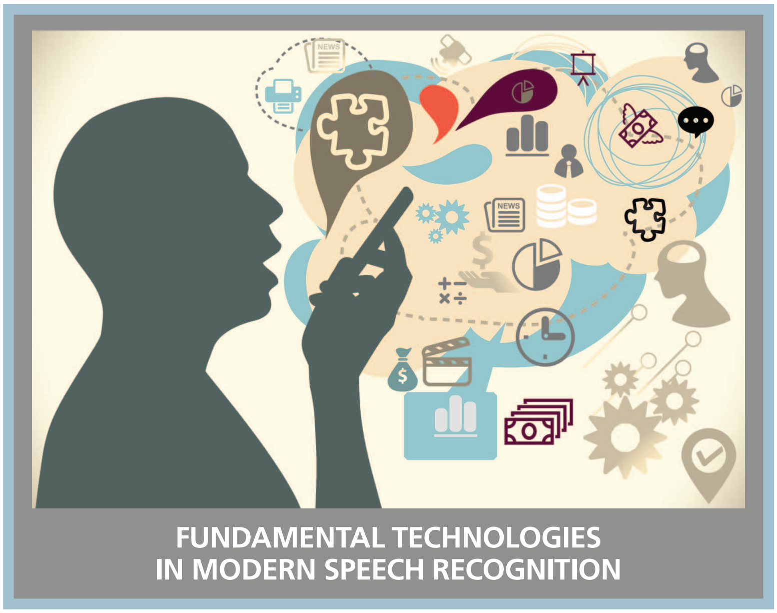
Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury

Deep Neural Networks for Acoustic Modeling in Speech Recognition

The shared views of four research groups



© ISTOCKPHOTO.COM/SUCHOA LERTADIPAT

Most current speech recognition systems use hidden Markov models (HMMs) to deal with the temporal variability of speech and Gaussian mixture models (GMMs) to determine how well each state of each HMM fits a frame or a short window of frames of coefficients that represents the acoustic input. An alternative way to evaluate the fit is to use a feed-forward neural network that takes several frames of coefficients as input and produces posterior probabilities over HMM states as output. Deep neural networks (DNNs) that have many hidden layers and are trained using new methods have been shown to outperform GMMs on a variety of speech recognition benchmarks, sometimes by a large margin. This article provides an overview of this progress and represents the shared views of four research groups that have had recent successes in using DNNs for acoustic modeling in speech recognition.

Digital Object Identifier 10.1109/MSP.2012.2205597
Date of publication: 15 October 2012

INTRODUCTION

New machine learning algorithms can lead to significant advances in automatic speech recognition (ASR). The biggest

1053-5888/12/$31.00©2012IEEE

single advance occurred nearly four decades ago with the introduction of the expectation-maximization (EM) algorithm for training HMMs (see [1] and [2] for informative historical reviews of the introduction of HMMs). With the EM algorithm, it became possible to develop speech recognition systems for real-world tasks using the richness of GMMs [3] to represent the relationship between HMM states and the acoustic input. In these systems the acoustic input is typically represented by concatenating Mel-frequency cepstral coefficients (MFCCs) or perceptual linear predictive coefficients (PLPs) [4] computed from the raw waveform and their first- and second-order temporal differences [5]. This nonadaptive but highly engineered preprocessing of the waveform is designed to discard the large amount of information in waveforms that is considered to be irrelevant for discrimination and to express the remaining information in a form that facilitates discrimination with GMM-HMMs.

DEEP NEURAL NETWORKS THAT HAVE MANY HIDDEN LAYERS AND ARE TRAINED USING NEW METHODS HAVE BEEN SHOWN TO OUTPERFORM GMMs ON A VARIETY OF SPEECH RECOGNITION BENCHMARKS, SOMETIMES BY A LARGE MARGIN.

GMMs have a number of advantages that make them suitable for modeling the probability distributions over vectors of input features that are associated with each state of an HMM. With enough components, they can model probability distributions to any required level of accuracy, and they are fairly easy to fit to data using the EM algorithm. A huge amount of research has gone into finding ways of constraining GMMs to increase their evaluation speed and to optimize the tradeoff between their flexibility and the amount of training data required to avoid serious overfitting [6].

The recognition accuracy of a GMM-HMM system can be further improved if it is discriminatively fine-tuned after it has been generatively trained to maximize its probability of generating the observed data, especially if the discriminative objective function used for training is closely related to the error rate on phones, words, or sentences [7]. The accuracy can also be improved by augmenting (or concatenating) the input features (e.g., MFCCs) with "tandem" or bottleneck features generated using neural networks [8], [69]. GMMs are so successful that it is difficult for any new method to outperform them for acoustic modeling.

Despite all their advantages, GMMs have a serious shortcoming—they are statistically inefficient for modeling data that lie on or near a nonlinear manifold in the data space. For example, modeling the set of points that lie very close to the surface of a sphere only requires a few parameters using an appropriate model class, but it requires a very large number of diagonal Gaussians or a fairly large number of full-covariance Gaussians. Speech is produced by modulating a relatively small number of parameters of a dynamical system [10], [11], and this implies that its true underlying structure is much lower-dimensional than is immediately apparent in a window that contains hundreds of coefficients. We believe, therefore, that other types of model may work better than GMMs for acoustic modeling if they can more effectively exploit information embedded in a large window of frames.

Artificial neural networks trained by backpropagating error derivatives have the potential to learn much better models of data that lie on or near a nonlinear manifold. In fact, two decades ago, researchers achieved some success using artificial neural networks with a single layer of nonlinear hidden units to predict HMM states from windows of acoustic coefficients [9]. At that time, however, neither the hardware nor the learning algorithms were adequate for training neural networks with many hidden layers on large amounts of data, and the performance benefits of using neural networks with a single hidden layer were not sufficiently large to seriously challenge GMMs. As a result, the main practical contribution of neural networks at that time was to provide extra features in tandem or bottleneck systems.

Over the last few years, advances in both machine learning algorithms and computer hardware have led to more efficient methods for training DNNs that contain many layers of nonlinear hidden units and a very large output layer. The large output layer is required to accommodate the large number of HMM states that arise when each phone is modeled by a number of different "triphone" HMMs that take into account the phones on either side. Even when many of the states of these triphone HMMs are tied together, there can be thousands of tied states. Using the new learning methods, several different research groups have shown that DNNs can outperform GMMs at acoustic modeling for speech recognition on a variety of data sets including large data sets with large vocabularies.

This review article aims to represent the shared views of research groups at the University of Toronto, Microsoft Research (MSR), Google, and IBM Research, who have all had recent successes in using DNNs for acoustic modeling. The article starts by describing the two-stage training procedure that is used for fitting the DNNs. In the first stage, layers of feature detectors are initialized, one layer at a time, by fitting a stack of generative models, each of which has one layer of latent variables. These generative models are trained without using any information about the HMM states that the acoustic model will need to discriminate. In the second stage, each generative model in the stack is used to initialize one layer of hidden units in a DNN and the whole network is then discriminatively fine-tuned to predict the target HMM states. These targets are obtained by using a baseline GMM-HMM system to produce a forced alignment.

In this article, we review exploratory experiments on the TIMIT database [12], [13] that were used to demonstrate the power of this two-stage training procedure for acoustic modeling. The DNNs that worked well on TIMIT were then applied to five different large-vocabulary continuous speech recognition (LVCSR) tasks by three different research groups whose results we also summarize. The DNNs worked well on all of these tasks when compared with highly tuned GMM-HMM systems, and on some of the tasks they outperformed the state of the art by a large margin. We also describe some other uses of DNNs for acoustic modeling and some variations on the training procedure.

OVER THE LAST FEW YEARS, ADVANCES IN BOTH MACHINE LEARNING ALGORITHMS AND COMPUTER HARDWARE HAVE LED TO MORE EFFICIENT METHODS FOR TRAINING DNNs.

TRAINING DEEP NEURAL NETWORKS

A DNN is a feed-forward, artificial neural network that has more than one layer of hidden units between its inputs and its outputs. Each hidden unit, j, typically uses the logistic function (the closely related hyberbolic tangent is also often used and any function with a well-behaved derivative can be used) to map its total input from the layer below, x_j, to the scalar state, y_j that it sends to the layer above.

$$y_j = \text{logistic}(x_j) = \frac{1}{1+e^{-x_j}}, \qquad x_j = b_j + \sum_i y_i w_{ij}, \quad (1)$$

where b_j is the bias of unit j, i is an index over units in the layer below, and w_{ij} is the weight on a connection to unit j from unit i in the layer below. For multiclass classification, output unit j converts its total input, x_j, into a class probability, p_j, by using the "softmax" nonlinearity

$$p_j = \frac{\exp(x_j)}{\sum_k \exp(x_k)}, \quad (2)$$

where k is an index over all classes.

DNNs can be discriminatively trained (DT) by backpropagating derivatives of a cost function that measures the discrepancy between the target outputs and the actual outputs produced for each training case [14]. When using the softmax output function, the natural cost function C is the cross entropy between the target probabilities d and the outputs of the softmax, p

$$C = -\sum_j d_j \log p_j, \quad (3)$$

where the target probabilities, typically taking values of one or zero, are the supervised information provided to train the DNN classifier.

For large training sets, it is typically more efficient to compute the derivatives on a small, random "minibatch" of training cases, rather than the whole training set, before updating the weights in proportion to the gradient. This stochastic gradient descent method can be further improved by using a "momentum" coefficient, $0 < \alpha < 1$, that smooths the gradient computed for minibatch t, thereby damping oscillations across ravines and speeding progress down ravines

$$\Delta w_{ij}(t) = \alpha \Delta w_{ij}(t-1) - \epsilon \frac{\partial C}{\partial w_{ij}(t)}. \quad (4)$$

The update rule for biases can be derived by treating them as weights on connections coming from units that always have a state of one.

To reduce overfitting, large weights can be penalized in proportion to their squared magnitude, or the learning can simply be terminated at the point at which performance on a held-out validation set starts getting worse [9]. In DNNs with full connectivity between adjacent layers, the initial weights are given small random values to prevent all of the hidden units in a layer from getting exactly the same gradient.

DNNs with many hidden layers are hard to optimize. Gradient descent from a random starting point near the origin is not the best way to find a good set of weights, and unless the initial scales of the weights are carefully chosen [15], the backpropagated gradients will have very different magnitudes in different layers. In addition to the optimization issues, DNNs may generalize poorly to held-out test data. DNNs with many hidden layers and many units per layer are very flexible models with a very large number of parameters. This makes them capable of modeling very complex and highly nonlinear relationships between inputs and outputs. This ability is important for high-quality acoustic modeling, but it also allows them to model spurious regularities that are an accidental property of the particular examples in the training set, which can lead to severe overfitting. Weight penalties or early stopping can reduce the overfitting but only by removing much of the modeling power. Very large training sets [16] can reduce overfitting while preserving modeling power, but only by making training very computationally expensive. What we need is a better method of using the information in the training set to build multiple layers of nonlinear feature detectors.

GENERATIVE PRETRAINING

Instead of designing feature detectors to be good for discriminating between classes, we can start by designing them to be good at modeling the structure in the input data. The idea is to learn one layer of feature detectors at a time with the states of the feature detectors in one layer acting as the data for training the next layer. After this generative "pretraining," the multiple layers of feature detectors can be used as a much better starting point for a discriminative "fine-tuning" phase during which backpropagation through the DNN slightly adjusts the weights found in pretraining [17]. Some of the high-level features created by the generative pretraining will be of little use for discrimination, but others will be far more useful than the raw inputs. The generative pretraining finds a region of the weight-space that allows the discriminative fine-tuning to make rapid progress, and it also significantly reduces overfitting [18].

A single layer of feature detectors can be learned by fitting a generative model with one layer of latent variables to the input data. There are two broad classes of generative model to choose from. A directed model generates data by first choosing the states of the latent variables from a prior distribution and then choosing the states of the observable variables from their conditional distributions given the latent states. Examples of directed models with one layer of latent variables are factor analysis, in which the latent variables are drawn from an isotropic Gaussian, and GMMs, in which they are drawn from a discrete distribution. An undirected model has a very different way of generating data. Instead of using one set of parameters to define a prior distribution over the latent variables and a separate set of parameters to define the conditional distributions of the observable variables given the values of the latent variables, an undirected model uses a single set of parameters, $\mathbf{W}$, to define the joint probability of a vector of values of the observable variables, $\mathbf{v}$, and a vector of values of the latent variables, $\mathbf{h}$, via an energy function, E

$$p(\mathbf{v},\mathbf{h};\mathbf{W}) = \frac{1}{Z}e^{-E(\mathbf{v},\mathbf{h};\mathbf{W})}, \quad Z = \sum_{\mathbf{v}',\mathbf{h}'} e^{-E(\mathbf{v}',\mathbf{h}';\mathbf{W})}, \tag{5}$$

where Z is called the partition function.

WHAT WE NEED IS A BETTER METHOD OF USING THE INFORMATION IN THE TRAINING SET TO BUILD MULTIPLE LAYERS OF NONLINEAR FEATURE DETECTORS.

If many different latent variables interact nonlinearly to generate each data vector, it is difficult to infer the states of the latent variables from the observed data in a directed model because of a phenomenon known as “explaining away” [19]. In undirected models, however, inference is easy provided the latent variables do not have edges linking them. Such a restricted class of undirected models is ideal for layerwise pretraining because each layer will have an easy inference procedure.

We start by describing an approximate learning algorithm for a restricted Boltzmann machine (RBM) which consists of a layer of stochastic binary “visible” units that represent binary input data connected to a layer of stochastic binary hidden units that learn to model significant nonindependencies between the visible units [20]. There are undirected connections between visible and hidden units but no visible-visible or hidden-hidden connections. An RBM is a type of Markov random field (MRF) but differs from most MRFs in several ways: it has a bipartite connectivity graph, it does not usually share weights between different units, and a subset of the variables are unobserved, even during training.

AN EFFICIENT LEARNING PROCEDURE FOR RBMs

A joint configuration, $(\mathbf{v},\mathbf{h})$ of the visible and hidden units of an RBM has an energy given by

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i\in\text{visible}} a_i v_i - \sum_{j\in\text{hidden}} b_j h_j - \sum_{i,j} v_i h_j w_{ij}, \tag{6}$$

where v_i, h_j are the binary states of visible unit i and hidden unit j, a_i, b_j are their biases, and w_{ij} is the weight between them. The network assigns a probability to every possible pair of a visible and a hidden vector via this energy function as in (5) and the probability that the network assigns to a visible vector, $\mathbf{v}$, is given by summing over all possible hidden vectors

$$p(\mathbf{v}) = \frac{1}{Z}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}. \tag{7}$$

The derivative of the log probability of a training set with respect to a weight is surprisingly simple

$$\frac{1}{N}\sum_{n=1}^{n=N}\frac{\partial \log p(\mathbf{v}^n)}{\partial w_{ij}} = <v_i h_j>_{\text{data}} - <v_i h_j>_{\text{model}}, \tag{8}$$

where N is the size of the training set and the angle brackets are used to denote expectations under the distribution specified by the subscript that follows. The simple derivative in (8) leads to a very simple learning rule for performing stochastic steepest ascent in the log probability of the training data

$$\Delta w_{ij} = \epsilon(<v_i h_j>_{\text{data}} - <v_i h_j>_{\text{model}}), \tag{9}$$

where ϵ is a learning rate.

The absence of direct connections between hidden units in an RBM makes it very easy to get an unbiased sample of $<v_i h_j>_{\text{data}}$. Given a randomly selected training case, $\mathbf{v}$, the binary state, h_j, of each hidden unit, j, is set to one with probability

$$p(h_j = 1|\mathbf{v}) = \text{logistic}(b_j + \sum_i v_i w_{ij}) \tag{10}$$

and $v_i h_j$ is then an unbiased sample. The absence of direct connections between visible units in an RBM makes it very easy to get an unbiased sample of the state of a visible unit, given a hidden vector

$$p(v_i = 1|h) = \text{logistic}(a_i + \sum_j h_j w_{ij}). \tag{11}$$

Getting an unbiased sample of $<v_i h_j>_{\text{model}}$, however, is much more difficult. It can be done by starting at any random state of the visible units and performing alternating Gibbs sampling for a very long time. Alternating Gibbs sampling consists of updating all of the hidden units in parallel using (10) followed by updating all of the visible units in parallel using (11).

A much faster learning procedure called contrastive divergence (CD) was proposed in [20]. This starts by setting the states of the visible units to a training vector. Then the binary states of the hidden units are all computed in parallel using (10). Once binary states have been chosen for the hidden units, a “reconstruction” is produced by setting each v_i to one with a probability given by (11). Finally, the states of the hidden units are updated again. The change in a weight is then given by

$$\Delta w_{ij} = \epsilon(<v_i h_j>_{\text{data}} - <v_i h_j>_{\text{recon}}). \tag{12}$$

A simplified version of the same learning rule that uses the states of individual units instead of pairwise products is used for the biases.

CD works well even though it is only crudely approximating the gradient of the log probability of the training data [20]. RBMs learn better generative models if more steps of alternating Gibbs sampling are used before collecting the statistics for the second term in the learning rule, but for the purposes of pretraining feature detectors, more alternations are generally of little value and all the results reviewed here were obtained using CD_1 which does a single full step of alternating Gibbs sampling after the initial update of the hidden units. To suppress noise in the learning, the real-valued probabilities rather than binary samples are generally used for the reconstructions and the subsequent states of the hidden units, but it is important to use sampled binary values for the first computation of the hidden states because the sampling noise acts as a very effective regularizer that prevents overfitting [21].

ONE VERY NICE PROPERTY OF A DBN THAT DISTINGUISHES IT FROM OTHER MULTILAYER, DIRECTED, NONLINEAR GENERATIVE MODELS IS THAT IT IS POSSIBLE TO INFER THE STATES OF THE LAYERS OF HIDDEN UNITS IN A SINGLE FORWARD PASS.

MODELING REAL-VALUED DATA

Real-valued data, such as MFCCs, are more naturally modeled by linear variables with Gaussian noise and the RBM energy function can be modified to accommodate such variables, giving a Gaussian–Bernoulli RBM (GRBM)

$$E(\mathbf{v},\mathbf{h}) = \sum_{i\in \text{vis}} \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{j\in \text{hid}} b_j h_j - \sum_{i,j} \frac{v_i}{\sigma_i} h_j w_{ij}, \tag{13}$$

where σ_i is the standard deviation of the Gaussian noise for visible unit i.

The two conditional distributions required for CD_1 learning are

$$p(h_j|\mathbf{v}) = \text{logistic}\left(b_j + \sum_i \frac{v_i}{\sigma_i} w_{ij}\right) \tag{14}$$

$$p(v_i|\mathbf{h}) = \mathcal{N}\left(a_i + \sigma_i \sum_j h_j w_{ij}, \sigma_i^2\right), \tag{15}$$

where $\mathcal{N}(\mu,\sigma^2)$ is a Gaussian. Learning the standard deviations of a GRBM is problematic for reasons described in [21], so for pretraining using CD_1, the data are normalized so that each coefficient has zero mean and unit variance, the standard deviations are set to one when computing $p(v|h)$, and no noise is added to the reconstructions. This avoids the issue of deciding the right noise level.

STACKING RBMs TO MAKE A DEEP BELIEF NETWORK

After training an RBM on the data, the inferred states of the hidden units can be used as data for training another RBM that learns to model the significant dependencies between the hidden units of the first RBM. This can be repeated as many times as desired to produce many layers of nonlinear feature detectors that represent progressively more complex statistical structure in the data. The RBMs in a stack can be combined in a surprising way to produce [22] a single, multilayer generative model called a deep belief net (DBN) (not to be confused with a dynamic Bayesian net, which is a type of directed model of temporal data that unfortunately has the same acronym). Even though each RBM is an undirected model, the DBN formed by the whole stack is a hybrid generative model whose top two layers are undirected (they are the final RBM in the stack) but whose lower layers have top-down, directed connections (see Figure 1).

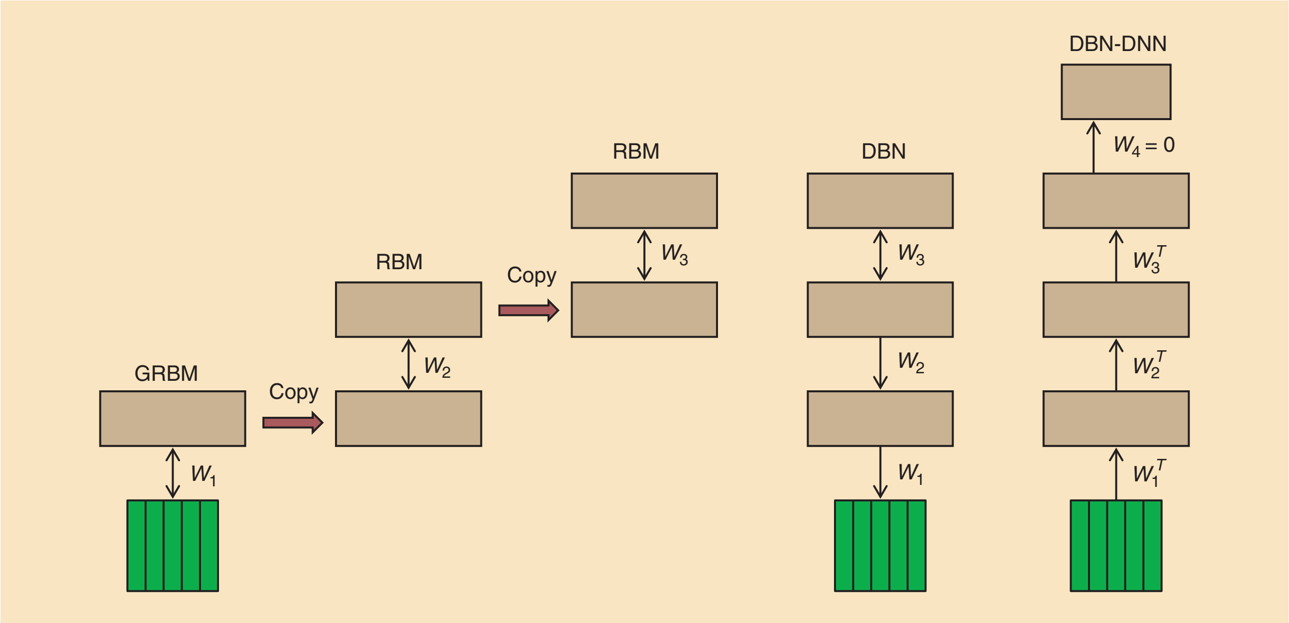
To understand how RBMs are composed into a DBN, it is helpful to rewrite (7) and to make explicit the dependence on $\mathbf{W}$:

$$p(\mathbf{v};\mathbf{W}) = \sum_h p(\mathbf{h};\mathbf{W})\, p(v|\mathbf{h};\mathbf{W}), \tag{16}$$

where $p(\mathbf{h};\mathbf{W})$ is defined as in (7) but with the roles of the visible and hidden units reversed. Now it is clear that the model can be improved by holding $p(\mathbf{v}|\mathbf{h};\mathbf{W})$ fixed after training the RBM, but replacing the prior over hidden vectors $p(\mathbf{h};\mathbf{W})$ by a better prior, i.e., a prior that is closer to the aggregated posterior over hidden vectors that can be sampled by first picking a training case and then inferring a hidden vector using (14). This aggregated posterior is exactly what the next RBM in the stack is trained to model.

As shown in [22], there is a series of variational bounds on the log probability of the training data, and furthermore, each time a new RBM is added to the stack, the variational bound on the new and deeper DBN is better than the previous variational bound, provided the new RBM is initialized and learned in the right way. While the existence of a bound that keeps improving is mathematically reassuring, it does not answer the practical issue, addressed in this article, of whether the learned feature detectors are useful for discrimination on a task that is unknown while training the DBN. Nor does it guarantee that anything improves when we use efficient short-cuts such as CD_1 training of the RBMs.

One very nice property of a DBN that distinguishes it from other multilayer, directed, nonlinear generative models is that it is possible to infer the states of the layers of hidden units in a single forward pass. This inference, which is used in deriving the variational bound, is not exactly correct but is fairly accurate. So after learning a DBN by training a stack of RBMs, we can jettison the whole probabilistic framework and simply use the generative weights in the reverse direction as a way of initializing all the feature detecting layers of a deterministic feed-forward DNN. We then just add a final softmax layer and train the whole DNN discriminatively. Unfortunately, a DNN that is pretrained generatively as a DBN is often still called a DBN in the literature. For clarity, we call it a DBN-DNN.



[FIG1] **The sequence of operations used to create a DBN with three hidden layers and to convert it to a pretrained DBN-DNN. First, a GRBM is trained to model a window of frames of real-valued acoustic coefficients. Then the states of the binary hidden units of the GRBM are used as data for training an RBM. This is repeated to create as many hidden layers as desired. Then the stack of RBMs is converted to a single generative model, a DBN, by replacing the undirected connections of the lower level RBMs by top-down, directed connections. Finally, a pretrained DBN-DNN is created by adding a "softmax" output layer that contains one unit for each possible state of each HMM. The DBN-DNN is then discriminatively trained to predict the HMM state corresponding to the central frame of the input window in a forced alignment.**

INTERFACING A DNN WITH AN HMM

After it has been discriminatively fine-tuned, a DNN outputs probabilities of the form $p(\text{HMMstate}|\text{AcousticInput})$. But to compute a Viterbi alignment or to run the forward-backward algorithm within the HMM framework, we require the likelihood $p(\text{AcousticInput}|\text{HMMstate})$. The posterior probabilities that the DNN outputs can be converted into the scaled likelihood by dividing them by the frequencies of the HMM states in the forced alignment that is used for fine-tuning the DNN [9]. All of the likelihoods produced in this way are scaled by the same unknown factor of $p(\text{AcousticInput})$, but this has no effect on the alignment. Although this conversion appears to have little effect on some recognition tasks, it can be important for tasks where training labels are highly unbalanced (e.g., with many frames of silences).

PHONETIC CLASSIFICATION AND RECOGNITION ON TIMIT

The TIMIT data set provides a simple and convenient way of testing new approaches to speech recognition. The training set is small enough to make it feasible to try many variations of a new method and many existing techniques have already been benchmarked on the core test set, so it is easy to see if a new approach is promising by comparing it with existing techniques that have been implemented by their proponents [23]. Experience has shown that performance improvements on TIMIT do not necessarily translate into performance improvements on large vocabulary tasks with less controlled recording conditions and much more training data. Nevertheless, TIMIT provides a good starting point for developing a new approach, especially one that requires a challenging amount of computation.

Mohamed et. al. [12] showed that a DBN-DNN acoustic model outperformed the best published recognition results on TIMIT at about the same time as Sainath et. al. [23] achieved a similar improvement on TIMIT by applying state-of-the-art techniques developed for large vocabulary recognition. Subsequent work combined the two approaches by using state-of-the-art, DT speaker-dependent features as input to the DBN-DNN [24], but this produced little further improvement, probably because the hidden layers of the DBN-DNN were already doing quite a good job of progressively eliminating speaker differences [25].

The DBN-DNNs that worked best on the TIMIT data formed the starting point for subsequent experiments on much more challenging large vocabulary tasks that were too computationally intensive to allow extensive exploration of variations in the architecture of the neural network, the representation of the acoustic input, or the training procedure.

For simplicity, all hidden layers always had the same size, but even with this constraint it was impossible to train all possible combinations of number of hidden layers [1, 2, **3, 4, 5, 6, 7, 8**], number of units per layer [512, **1,024, 2,048, 3,072**], and number of frames of acoustic data in the input layer [7, **11, 15, 17**, 27, 37]. Fortunately, the performance of the networks on the TIMIT core test set was fairly insensitive to the precise details of the architecture and the results in [13] suggest that any combination of the numbers in boldface probably has an error rate within about 2% of the very best combination. This robustness is crucial for methods such as DBN-DNNs that have a lot of tuneable metaparameters. Our consistent finding is that multiple hidden layers always worked better than one hidden layer and, with multiple hidden layers, pretraining always improved the results on both the development and test sets in the TIMIT task. Details of the learning rates, stopping criteria, momentum, L2 weight penalties and minibatch size for both the pretraining and fine-tuning are given in [13].

Table 1 compares DBN-DNNs with a variety of other methods on the TIMIT core test set. For each type of DBN-DNN the architecture that performed best on the development set is reported. All methods use MFCCs as inputs except for the three marked “fbank” that use log Mel-scale filter-bank outputs.

PREPROCESSING THE WAVEFORM FOR DEEP NEURAL NETWORKS

State-of-the-art ASR systems do not use filter-bank coefficients as the input representation because they are strongly correlated so modeling them well requires either full covariance Gaussians or a huge number of diagonal Gaussians. MFCCs offer a more suitable alternative as their individual components are roughly independent so they are much easier to model using a mixture of diagonal covariance Gaussians. DBN-DNNs do not require uncorrelated data and, on the TIMIT database, the work reported in [13] showed that the best performing DBN-DNNs trained with filter-bank features had a phone error rate 1.7% lower than the best performing DBN-DNNs trained with MFCCs (see Table 1).

FINE-TUNING DBN-DNNs TO OPTIMIZE MUTUAL INFORMATION

In the experiments using TIMIT discussed above, the DNNs were fine-tuned to optimize the per frame cross entropy between the target HMM state and the predictions. The transition parameters and language model scores were obtained from an HMM-like approach and were trained independently of the DNN weights. However, it has long been known that sequence classification criteria, which are more directly correlated with the overall word or phone error rate, can be very helpful in improving recognition accuracy [7], [35] and the benefit of using such sequence classification criteria with shallow neural networks has already been shown by [36]–[38]. In the more recent work reported in [31], one popular type of sequence classification criterion, maximum mutual information (MMI), proposed as early as 1986 [7], was successfully applied to learn DBN-DNN weights for the TIMIT phone recognition task. MMI optimizes the conditional probability $p(l_{1:T}|v_{1:T})$ of the whole sequence of labels, $l_{1:T}$, with length T, given the whole visible feature utterance $v_{1:T}$, or equivalently the hidden feature sequence $h_{1:T}$ extracted by the DNN

$$p(l_{1:T}|v_{1:T}) = p(l_{1:T}|h_{1:T}) = \frac{\exp\left(\sum_{t=1}^{T}\gamma_{ij}\phi_{ij}(l_{t-1},l_t) + \sum_{t=1}^{T}\sum_{d=1}^{D}\lambda_{l_t,d}h_{td}\right)}{Z(h_{1:T})}, \tag{17}$$

where the transition feature $\phi_{ij}(l_{t-1},l_t)$ takes on a value of one if $l_{t-1}=i$ and $l_t=j$, and otherwise takes on a value of zero, where γ_{ij} is the parameter associated with this transition feature, h_{td} is the dth dimension of the hidden unit value at the tth frame at the final layer of the DNN, and where D is the number of units in the final hidden layer. Note the objective function of (17) derived from mutual information [35] is the same as the conditional likelihood associated with a specialized linear-chain conditional random field. Here, it is the topmost layer of the DNN below the softmax layer, not the raw speech coefficients of MFCC or PLP, that provides “features” to the conditional random field.

[TABLE 1] COMPARISONS AMONG THE REPORTED SPEAKER-INDEPENDENT (SI) PHONETIC RECOGNITION ACCURACY RESULTS ON TIMIT CORE TEST SET WITH 192 SENTENCES.

METHOD	PER
CD-HMM [26]	27.3%
AUGMENTED CONDITIONAL RANDOM FIELDS [26]	26.6%
RANDOMLY INITIALIZED RECURRENT NEURAL NETS [27]	26.1%
BAYESIAN TRIPHONE GMM-HMM [28]	25.6%
MONOPHONE HTMS [29]	24.8%
HETEROGENEOUS CLASSIFIERS [30]	24.4%
MONOPHONE RANDOMLY INITIALIZED DNNs (SIX LAYERS) [13]	23.4%
MONOPHONE DBN-DNNs (SIX LAYERS) [13]	22.4%
MONOPHONE DBN-DNNs WITH MMI TRAINING [31]	22.1%
TRIPHONE GMM-HMMs DT W/ BMMI [32]	21.7%
MONOPHONE DBN-DNNs ON FBANK (EIGHT LAYERS) [13]	20.7%
MONOPHONE MCRBM-DBN-DNNs ON FBANK (FIVE LAYERS) [33]	20.5%
MONOPHONE CONVOLUTIONAL DNNs ON FBANK (THREE LAYERS) [34]	20.0%

To optimize the log conditional probability $p(l^n_{1:T}|v^n_{1:T})$ of the nth utterance, we take the gradient over the activation parameters λ_{kd}, transition parameters γ_{ij}, and the lower-layer weights of the DNN, w_{ij}, according to

$$\frac{\partial \log p(l^n_{1:T}|v^n_{1:T})}{\partial \lambda_{kd}} = \sum_{t=1}^{T}(\delta(l^n_t = k) - p(l^n_t = k|v^n_{1:T}))h^n_{td} \tag{18}$$

$$\frac{\partial \log p(l^n_{1:T}|v^n_{1:T})}{\partial \gamma_{ij}} = \sum_{t=1}^{T}[\delta(l^n_{t-1} = i, l^n_t = j) - p(l^n_{t-1} = i, l^n_t = j|v^n_{1:T})] \tag{19}$$

$$\frac{\partial \log p(l^n_{1:T}|v^n_{1:T})}{\partial w_{ij}} = \sum_{t=1}^{T}\left[\lambda_{l_{td}} - \sum_{k=1}^{K} p(l^n_t = k|v^n_{1:T})\lambda_{kd}\right] \times h^n_{td}(1-h^n_{td})x^n_{ti}. \tag{20}$$

Note that the gradient $(\partial \log p(l^n_{1:T}|v^n_{1:T}))/(\partial w_{ij})$ above can be viewed as back-propagating the error $\delta(l^n_t = k) - p(l^n_t = k|v^n_{1:T})$, versus $\delta(l^n_t = k) - p(l^n_t = k|v^n_t)$ in the frame-based training algorithm.

In implementing the above learning algorithm for a DBN-DNN, the DNN weights can first be fine-tuned to optimize the per frame cross entropy. The transition parameters can be initialized from the combination of the HMM transition matrices and the "phone language" model scores, and can be further optimized by tuning the transition features while fixing the DNN weights before the joint optimization. Using the joint optimization with careful scheduling, we observe that the sequential MMI training can outperform the frame-level training by about 5% relative within the same system in the same laboratory.

CONVOLUTIONAL DNNs FOR PHONE CLASSIFICATION AND RECOGNITION

All the previously cited work reported phone recognition results on the TIMIT database. In recognition experiments, the input is the acoustic input for the whole utterance while the output is the spoken phonetic sequence. A decoding process using a phone language model is used to produce this output sequence. Phonetic classification is a different task where the acoustic input has already been labeled with the correct boundaries between different phonetic units and the goal is to classify these phones conditioned on the given boundaries. In [39], convolutional DBN-DNNs were introduced and successfully applied to various audio tasks including phone classification on the TIMIT database. In this model, the RBM was made convolutional in time by sharing weights between hidden units that detect the same feature at different times. A max-pooling operation was then performed, which takes the maximal activation over a pool of adjacent hidden units that share the same weights but apply them at different times. This yields some temporal invariance.

THE SUCCESS OF DBN-DNNs ON TIMIT TASKS STARTING IN 2009 MOTIVATED MORE AMBITIOUS EXPERIMENTS WITH MUCH LARGER VOCABULARIES AND MORE VARIED SPEAKING STYLES.

Although convolutional models along the temporal dimension achieved good classification results [39], applying them to phone recognition is not straightforward. This is because temporal variations in speech can be partially handled by the dynamic programing procedure in the HMM component and those aspects of temporal variation that cannot be adequately handled by the HMM can be addressed more explicitly and effectively by hidden trajectory models [40].

The work reported in [34] applied local convolutional filters with max-pooling to the frequency rather than time dimension of the spectrogram. Sharing-weights and pooling over frequency was motivated by the shifts in formant frequencies caused by speaker variations. It provides some speaker invariance while also offering noise robustness due to the band-limited nature of the filters. [34] only used weight-sharing and max-pooling across nearby frequencies because, unlike features that occur at different positions in images, acoustic features occurring at very different frequencies are very different.

A SUMMARY OF THE DIFFERENCES BETWEEN DNNs AND GMMs

Here we summarize the main differences between the DNNs and GMMs used in the TIMIT experiments described so far in this article. First, one major element of the DBN-DNN, the RBM, which serves as the building block for pretraining, is an instance of "product of experts" [20], in contrast to mixture models that are a "sum of experts." Product models have only very recently been explored in speech processing, e.g., [41]. Mixture models with a large number of components use their parameters inefficiently because each parameter only applies to a very small fraction of the data whereas each parameter of a product model is constrained by a large fraction of the data. Second, while both DNNs and GMMs are nonlinear models, the nature of the nonlinearity is very different. A DNN has no problem modeling multiple simultaneous events within one frame or window because it can use different subsets of its hidden units to model different events. By contrast, a GMM assumes that each datapoint is generated by a single component of the mixture so it has no efficient way of modeling multiple simultaneous events. Third, DNNs are good at exploiting multiple frames of input coefficients whereas GMMs that use diagonal covariance matrices benefit much less from multiple frames because they require decorrelated inputs. Finally, DNNs are learned using stochastic gradient descent, while GMMs are learned using the EM algorithm or its extensions [35], which makes GMM learning much easier to parallelize on a cluster machine.

COMPARING DBN-DNNs WITH GMMs FOR LARGE-VOCABULARY SPEECH RECOGNITION

The success of DBN-DNNs on TIMIT tasks starting in 2009 motivated more ambitious experiments with much larger vocabularies and more varied speaking styles. In this section, we review experiments by three different speech groups on five different benchmark tasks for large-vocabulary speech recognition. To make DBN-DNNs work really well on large vocabulary tasks it is important to replace the monophone HMMs used for TIMIT (and also for early neural network/HMM hybrid systems) with triphone HMMs that have many thousands of tied states [42]. Predicting these context-dependent states provides several advantages over monophone targets. They supply more bits of information per frame in the labels. They also make it possible to use a more powerful triphone HMM decoder and to exploit the sensible classes discovered by the decision tree clustering that is used to tie the states of different triphone HMMs. Using context-dependent HMM states, it is possible to outperform state-of-the-art BMMI trained GMM-HMM systems with a two-hidden-layer neural network without using any pretraining [43], though using more hidden layers and pretraining works even better.

BING-VOICE-SEARCH SPEECH RECOGNITION TASK

The first successful use of acoustic models based on DBN-DNNs for a large vocabulary task used data collected from the Bing mobile voice search application (BMVS). The task used 24 h of training data with a high degree of acoustic variability caused by noise, music, side-speech, accents, sloppy pronunciation, hesitation, repetition, interruptions, and mobile phone differences. The results reported in [42] demonstrated that the best DNN-HMM acoustic model trained with context-dependent states as targets achieved a sentence accuracy of 69.6% on the test set, compared with 63.8% for a strong, minimum phone error (MPE)-trained GMM-HMM baseline.

The DBN-DNN used in the experiments was based on one of the DBN-DNNs that worked well for the TIMIT task. It used five pretrained layers of hidden units with 2,048 units per layer and was trained to classify the central frame of an 11-frame acoustic context window using 761 possible context-dependent states as targets. In addition to demonstrating that a DBN-DNN could provide gains on a large vocabulary task, several other important issues were explicitly investigated in [42]. It was found that using tied triphone context-dependent state targets was crucial and clearly superior to using monophone state targets, even when the latter were derived from the same forced alignment with the same baseline. It was also confirmed that the lower the error rate of the system used during forced alignment to generate frame-level training labels for the neural net, the lower the error rate of the final neural-net-based system. This effect was consistent across all the alignments they tried, including monophone alignments, alignments from ML-trained GMM-HMM systems, and alignments from DT GMM-HMM systems.

Further work after that of [42] extended the DNN-HMM acoustic model from 24 h of training data to 48 h and explored the respective roles of pretraining and fine-tuning the DBN-DNN [44]. As expected, pretraining is helpful in training the DBN-DNN because it initializes the DBN-DNN weights to a point in the weight-space from which fine-tuning is highly effective. However, a moderate increase of the amount of unlabeled pretraining data has an insignificant effect on the final recognition results (69.6% to 69.8%), as long as the original training set is fairly large. By contrast, the same amount of additional labeled fine-tuning training data significantly improves the performance of the DNN-HMMs (accuracy from 69.6% to 71.7%).

[TABLE 2] COMPARING FIVE DIFFERENT DBN-DNN ACOUSTIC MODELS WITH TWO STRONG GMM-HMM BASELINE SYSTEMS THAT ARE DISCRIMINATIVELY TRAINED. SI TRAINING ON 309 H OF DATA AND SINGLE-PASS DECODING WERE USED FOR ALL MODELS EXCEPT FOR THE GMM-HMM SYSTEM SHOWN ON THE LAST ROW WHICH USED SA TRAINING WITH 2,000 H OF DATA AND MULTIPASS DECODING INCLUDING HYPOTHESES COMBINATION. IN THE TABLE, "40 MIX" MEANS A MIXTURE OF 40 GAUSSIANS PER HMM STATE AND "15.2 NZ" MEANS 15.2 MILLION, NONZERO WEIGHTS. WERs IN % ARE SHOWN FOR TWO SEPARATE TEST SETS, HUB500-SWB AND RT03S-FSH.

		WER	
MODELING TECHNIQUE	**#PARAMS [10^6]**	**HUB5'00-SWB**	**RT03S-FSH**
GMM, 40 MIX DT 309H SI	29.4	23.6	27.4
NN 1 HIDDEN-LAYER × 4,634 UNITS	43.6	26.0	29.4
+ 2 × 5 NEIGHBORING FRAMES	45.1	22.4	25.7
DBN-DNN 7 HIDDEN LAYERS × 2,048 UNITS	45.1	17.1	19.6
+ UPDATED STATE ALIGNMENT	45.1	16.4	18.6
+ SPARSIFICATION	15.2 NZ	16.1	18.5
GMM 72 MIX DT 2000H SA	102.4	17.1	18.6

SWITCHBOARD SPEECH RECOGNITION TASK

The DNN-HMM training recipe developed for the Bing voice search data was applied unaltered to the Switchboard speech recognition task [43] to confirm the suitability of DNN-HMM acoustic models for large vocabulary tasks. Before this work, DNN-HMM acoustic models had only been trained with up to 48 h of data [44] and hundreds of tied triphone states as targets, whereas this work used over 300 h of training data and thousands of tied triphone states as targets. Furthermore, Switchboard is a publicly available speech-to-text transcription benchmark task that allows much more rigorous comparisons among techniques.

The baseline GMM-HMM system on the Switchboard task was trained using the standard 309-h Switchboard-I training set. Thirteen-dimensional PLP features with windowed mean-variance normalization were concatenated with up to third-order derivatives and reduced to 39 dimensions by a form of linear discriminant analysis (LDA) called heteroscedastic LDA (HDLA). The SI crossword triphones used the common left-to-right three-state topology and shared 9,304 tied states.

The baseline GMM-HMM system had a mixture of 40 Gaussians per (tied) HMM state that were first trained generatively to optimize a maximum likelihood (ML) criterion and then refined discriminatively to optimize a boosted maximum-mutual-information (BMMI) criterion. A seven-hidden-layer DBN-DNN with 2,048 units in each layer and full connectivity between adjacent layers replaced the GMM in the acoustic model. The trigram language model, used for both systems, was trained on the training transcripts of the 2,000 h of the Fisher corpus and interpolated with a trigram model trained on written text.

The primary test set is the FSH portion of the 6.3-h Spring 2003 National Institute of Standards and Technology rich transcription set (RT03S). Table 2 extracted from the literature shows a summary of the core results. Using a DNN reduced the word error rate (WER) from the 27.4% of the baseline GMM-HMM (trained with BMMI) to 18.5%—a 33% relative reduction. The DNN-HMM system trained on 309 h performs as well as combining several speaker-adaptive (SA), multipass systems that use vocal tract length normalization (VTLN) and nearly seven times as much acoustic training data (the 2,000-h Fisher corpus) (18.6%; see the last row in Table 2).

Detailed experiments [43] on the Switchboard task confirmed that the remarkable accuracy gains from the DNN-HMM acoustic model are due to the direct modeling of tied triphone states using the DBN-DNN, the effective exploitation of neighboring frames by the DBN-DNN, and the strong modeling power of deeper networks, as was discovered in the Bing voice search task [44], [42]. Pretraining the DBN-DNN leads to the best results but it is not critical: For this task, it provides an absolute WER reduction of less than 1% and this gain is even smaller when using five or more hidden layers. For underresourced languages that have smaller amounts of labeled data, pretraining is likely to be far more helpful.

Further study [45] suggests that feature-engineering techniques such as HLDA and VTLN, commonly used in GMM-HMMs, are more helpful for shallow neural nets than for DBN-DNNs, presumably because DBN-DNNs are able to learn appropriate features in their lower layers.

GOOGLE VOICE INPUT SPEECH RECOGNITION TASK

Google Voice Input transcribes voice search queries, short messages, e-mails, and user actions from mobile devices. This is a large vocabulary task that uses a language model designed for a mixture of search queries and dictation.

Google's full-blown model for this task, which was built from a very large corpus, uses an SI GMM-HMM model composed of context-dependent crossword triphone HMMs that have a left-to-right, three-state topology. This model has a total of 7,969 senone states and uses as acoustic input PLP features that have been transformed by LDA. Semi-tied covariances (STCs) are used in the GMMs to model the LDA transformed features and BMMI [46] was used to train the model discriminatively.

[PRETRAINING DNNs AS GENERATIVE MODELS LED TO BETTER RECOGNITION RESULTS ON TIMIT AND SUBSEQUENTLY ON A VARIETY OF LVCSR TASKS.]

Jaitly et. al. [47] used this model to obtain approximately 5,870 h of aligned training data for a DBN-DNN acoustic model that predicts the 7,969 HMM state posteriors from the acoustic input. The DBN-DNN was loosely based on one of the DBN-DNNs used for the TIMIT task. It had four hidden layers with 2,560 fully connected units per layer and a final "softmax" layer with 7,969 alternative states. Its input was 11 contiguous frames of 40 log filter-bank outputs with no temporal derivatives. Each DBN-DNN layer was pretrained for one epoch as an RBM and then the resulting DNN was discriminatively fine-tuned for one epoch. Weights with magnitudes below a threshold were then permanently set to zero before a further quarter epoch of training. One third of the weights in the final network were zero. In addition to the DBN-DNN training, sequence-level discriminative fine-tuning of the neural network was performed using MMI, similar to the method proposed in [37]. Model combination was then used to combine results from the GMM-HMM system with the DNN-HMM hybrid, using the segmental conditional random field (SCARF) framework [47]. Viterbi decoding was done using the Google system [48] with modifications to compute the scaled log likelihoods from the estimates of the posterior probabilities and the state priors. Unlike the other systems, it was observed that for Voice Input it was essential to smooth the estimated priors for good performance. This smoothing of the priors was performed by rescaling the log priors with a multiplier that was chosen by using a grid search to find a joint optimum of the language model weight, the word insertion penalty, and the smoothing factor.

On a test set of anonymized utterances from the live Voice Input system, the DBN-DNN-based system achieved a WER of 12.3%—a 23% relative reduction compared to the best GMM-based system for this task. MMI sequence discriminative training gave an error rate of 12.2% and model combination with the GMM system 11.8%.

YOUTUBE SPEECH RECOGNITION TASK

In this task, the goal is to transcribe YouTube data. Unlike the mobile voice input applications described above, this application does not have a strong language model to constrain the interpretation of the acoustic information so good discrimination requires an accurate acoustic model.

Google's full-blown baseline, built with a much larger training set, was used to create approximately 1,400 h of aligned training data. This was used to create a new baseline system for which the input was nine frames of MFCCs that were transformed by LDA. SA training was performed, and decision tree clustering was used to obtain 17,552 triphone states. STCs were used in the GMMs to model the features. The acoustic models were further improved with BMMI. During decoding, ML linear regression (MLLR) and feature space MLLR (fMLLR) transforms were applied.

The acoustic data used for training the DBN-DNN acoustic model were the fMLLR-transformed features. The large number of HMM states added significantly to the computational burden, since most of the computation is done at the output layer. To reduce this burden, the DNN used only four hidden layers with 2,000 units in the first hidden layer and only 1,000 units in each of the layers above.

About ten epochs of training were performed on this data before sequence-level training and model combination. The DBN-DNN gave an absolute improvement of 4.7% over the baseline system's WER of 52.3%. Sequence-level fine-tuning of the DBN-DNN further improved results by 0.5% and model combination produced an additional gain of 0.9%.

ENGLISH BROADCAST NEWS SPEECH RECOGNITION TASK

DNNs have also been successfully applied to an English broadcast news task. Since a GMM-HMM baseline creates the initial training labels for the DNN, it is important to have a good baseline system. All GMM-HMM systems created at IBM use the following recipe to produce a state-of-the-art baseline system. First, SI features are created, followed by SA-trained (SAT) and DT features. Specifically, given initial PLP features, a set of SI features are created using LDA. Further processing of LDA features is performed to create SAT features using VTLN followed by fMLLR. Finally, feature and model-space discriminative training is applied using the BMMI or MPE criterion.

Using alignments from a baseline system, [32] trained a DBN-DNN acoustic model on 50 h of data from the 1996 and 1997 English Broadcast News Speech Corpora [37]. The DBN-DNN was trained with the best-performing LVCSR features, specifically the SAT+DT features. The DBN-DNN architecture consisted of six hidden layers with 1,024 units per layer and a final softmax layer of 2,220 context-dependent states. The SAT+DT feature input into the first layer used a context of nine frames. Pretraining was performed following a recipe similar to [42].

[DISCRIMINATIVE PRETRAINING HAS ALSO BEEN FOUND EFFECTIVE FOR THE ARCHITECTURES CALLED "DEEP CONVEX NETWORK" AND "DEEP STACKING NETWORK," WHERE PRETRAINING IS ACCOMPLISHED BY CONVEX OPTIMIZATION INVOLVING NO GENERATIVE MODELS.]

Two phases of fine-tuning were performed. During the first phase, the cross entropy loss was used. For cross entropy training, after each iteration through the whole training set, loss is measured on a held-out set and the learning rate is annealed (i.e., reduced) by a factor of two if the held-out loss has grown or improves by less than a threshold of 0.01% from the previous iteration. Once the learning rate has been annealed five times, the first phase of fine-tuning stops. After weights are learned via cross entropy, these weights are used as a starting point for a second phase of fine-tuning using a sequence criterion [37] that utilizes the MPE objective function, a discriminative objective function similar to MMI [7] but which takes into account phoneme error rate.

A strong SAT+DT GMM-HMM baseline system, which consisted of 2,220 context-dependent states and 50,000 Gaussians, gave a WER of 18.8% on the EARS Dev-04f set, whereas the DNN-HMM system gave 17.5% [50].

SUMMARY OF THE MAIN RESULTS FOR DBN-DNN ACOUSTIC MODELS ON LVCSR TASKS

Table 3 summarizes the acoustic modeling results described above. It shows that DNN-HMMs consistently outperform GMM-HMMs that are trained on the same amount of data, sometimes by a large margin. For some tasks, DNN-HMMs also outperform GMM-HMMs that are trained on much more data.

[TABLE 3] A COMPARISON OF THE PERCENTAGE WERs USING DNN-HMMs AND GMM-HMMs ON FIVE DIFFERENT LARGE VOCABULARY TASKS.

TASK	HOURS OF TRAINING DATA	DNN-HMM	GMM-HMM WITH SAME DATA	GMM-HMM WITH MORE DATA
SWITCHBOARD (TEST SET 1)	309	18.5	27.4	18.6 (2,000 H)
SWITCHBOARD (TEST SET 2)	309	16.1	23.6	17.1 (2,000 H)
ENGLISH BROADCAST NEWS	50	17.5	18.8	
BING VOICE SEARCH (SENTENCE ERROR RATES)	24	30.4	36.2	
GOOGLE VOICE INPUT	5,870	12.3		16.0 (>> 5,870 H)
YOUTUBE	1,400	47.6	52.3	

SPEEDING UP DNNs AT RECOGNITION TIME

State pruning or Gaussian selection methods can be used to make GMM-HMM systems computationally efficient at recognition time. A DNN, however, uses virtually all its parameters at every frame to compute state likelihoods, making it potentially much slower than a GMM with a comparable number of parameters. Fortunately, the time that a DNN-HMM system requires to recognize 1 s of speech can be reduced from 1.6 s to 210 ms, without decreasing recognition accuracy, by quantizing the weights down to 8 b and using the very fast SIMD primitives for fixed-point computation that are provided by a modern x86 central processing unit [49]. Alternatively, it can be reduced to 66 ms by using a graphics processing unit (GPU).

ALTERNATIVE PRETRAINING METHODS FOR DNNs

Pretraining DNNs as generative models led to better recognition results on TIMIT and subsequently on a variety of LVCSR tasks. Once it was shown that DBN-DNNs could learn good acoustic models, further research revealed that they could be trained in many different ways. It is possible to learn a DNN by starting with a shallow neural net with a single hidden layer. Once this net has been trained discriminatively, a second hidden layer is interposed between the first hidden layer and the softmax output units and the whole network is again discriminatively trained. This can be continued until the desired number of hidden layers is reached, after which full backpropagation fine-tuning is applied.

This type of discriminative pretraining works well in practice, approaching the accuracy achieved by generative DBN pretraining and further improvement can be achieved by stopping the discriminative pretraining after a single epoch instead of multiple epochs as reported in [45]. Discriminative pretraining has also been found effective for the architectures called "deep convex network" [51] and "deep stacking network" [52], where pretraining is accomplished by convex optimization involving no generative models.

Purely discriminative training of the whole DNN from random initial weights works much better than had been thought, provided the scales of the initial weights are set carefully, a large amount of labeled training data is available, and minibatch sizes over training epochs are set appropriately [45], [53]. Nevertheless, generative pretraining still improves test performance, sometimes by a significant amount.

Layer-by-layer generative pretraining was originally done using RBMs, but various types of autoencoder with one hidden layer can also be used (see Figure 2). On vision tasks, performance similar to RBMs can be achieved by pretraining with “denoising” autoencoders [54] that are regularized by setting a subset of the inputs to zero or “contractive” autoencoders [55] that are regularized by penalizing the gradient of the activities of the hidden units with respect to the inputs. For speech recognition, improved performance was achieved on both TIMIT and Broadcast News tasks by pretraining with a type of autoencoder that tries to find sparse codes [56].

> THE FINE-TUNING OF DNN ACOUSTIC MODELS IS TYPICALLY STOPPED EARLY TO PREVENT OVERFITTING, AND IT IS NOT CLEAR THAT THE MORE SOPHISTICATED METHODS ARE WORTHWHILE FOR SUCH INCOMPLETE OPTIMIZATION.

ALTERNATIVE FINE-TUNING METHODS FOR DNNs

Very large GMM acoustic models are trained by making use of the parallelism available in compute clusters. It is more difficult to use the parallelism of cluster systems effectively when training DBN-DNNs. At present, the most effective parallelization method is to parallelize the matrix operations using a GPU. This gives a speed-up of between one and two orders of magnitude, but the fine-tuning stage remains a serious bottleneck, and more effective ways of parallelizing training are needed. Some recent attempts are described in [52] and [57].

Most DBN-DNN acoustic models are fine-tuned by applying stochastic gradient descent with momentum to small minibatches of training cases. More sophisticated optimization methods that can be used on larger minibatches include nonlinear conjugate-gradient [17], LBFGS [58], and “Hessian-free” methods adapted to work for DNNs [59]. However, the fine-tuning of DNN acoustic models is typically stopped early to prevent overfitting, and it is not clear that the more sophisticated methods are worthwhile for such incomplete optimization.

OTHER WAYS OF USING DEEP NEURAL NETWORKS FOR SPEECH RECOGNITION

The previous section reviewed experiments in which GMMs were replaced by DBN-DNN acoustic models to give hybrid DNN-HMM systems in which the posterior probabilities over HMM states produced by the DBN-DNN replace the GMM output model. In this section, we describe two other ways of using DNNs for speech recognition.

USING DBN-DNNs TO PROVIDE INPUT FEATURES FOR GMM-HMM SYSTEMS

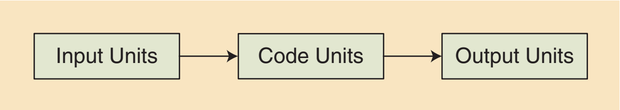
Here we describe a class of methods where neural networks are used to provide the feature vectors that the GMM in a GMM-HMM system is trained to model. The most common approach to extracting these feature vectors is to discriminatively train a randomly initialized neural net with a narrow bottleneck middle layer and to use the activations of the bottleneck hidden units as features. For a summary of such methods, commonly known as the tandem approach, see [60], [61], and [63].

Instead of replacing the coefficients usually modeled by GMMs, neural networks can also be used to provide additional features for the GMM to model [8], [9], [63]. DBN-DNNs have recently been shown to be very effective in such tandem systems. On the Aurora2 test set, pretraining decreased WERs by more than one third for speech with signal-to-noise levels of 20 dB or more, though this effect almost disappeared for very high noise levels [64].

Recently, [62] investigated a less direct way of producing feature vectors for the GMM. First, a DNN with six hidden layers of 1,024 units each was trained to achieve good classification accuracy for the 384 HMM states represented in its softmax output layer. This DNN did not have a bottleneck layer and was therefore able to classify better than a DNN with a bottleneck. Then the 384 logits computed by the DNN as input to its softmax layer were compressed down to 40 values using a 384-128-40-384 autoencoder. This method of producing feature vectors is called AE-BN because the bottleneck is in the autoencoder rather than in the DNN that is trained to classify HMM states.

Bottleneck feature experiments were conducted on 50-h and 430-h of data from the 1996 and 1997 English Broadcast News Speech collections and English broadcast audio from TDT-4. The baseline GMM-HMM acoustic model trained on 50 h was the same acoustic model described in the section “English Broadcast News Speech Recognition Task.” The acoustic model trained on 430-h had 6,000 states and 150,000 Gaussians. Again, the standard IBM LVCSR recipe described in the aforementioned section was used to create a set of SA DT features and models.

All DBN-DNNs used SAT features as input. They were pretrained as DBNs and then discriminatively fine-tuned to predict target values for 384 HMM states that were obtained by clustering the context-dependent states in the baseline GMM-HMM system. As in the section “English Broadcast News Speech Recognition Task,” the DBN-DNN was trained using the cross entropy criterion, followed by the sequence criterion with the same annealing and stopping rules.



[FIG2] An autoencoder is trained to minimize the discrepancy between the input vector and its reconstruction of the input vector on its output units. If the code units and the output units are both linear and the discrepancy is the squared reconstruction error, an autoencoder finds the same solution as principal components analysis (PCA) (up to a rotation of the components). If the output units and the code units are logistic, an autoencoder is quite similar to an RBM that is trained using CD, but it does not work as well for pretraining DNNs unless it is strongly regularized in an appropriate way. If extra hidden layers are added before and/or after the code layer, an autoencoder can compress data much better than PCA [17].

After the training of the first DBN-DNN terminated, the final set of weights was used for generating the 384 logits at the output layer. A second 384-128-40-384 DBN-DNN was then trained as an autoencoder to reduce the dimensionality of the output logits. The GMM-HMM system that used the feature vectors produced by the AE-BN was trained using feature and model space discriminative training. Both pretraining and the use of deeper networks made the AE-BN features work better for recognition. To fairly compare the performance of the system that used the AE-BN features with the baseline GMM-HMM system, the acoustic model of the AE-BN features was trained with the same number of states and Gaussians as the baseline system.

Table 4 shows the results of the AE-BN and baseline systems on both 50- and 430-h, for different steps in the LVCSR recipe described in the section "English Broadcast News Speech Recognition Task." On 50-h, the AE-BN system offers a 1.3% absolute improvement over the baseline GMM-HMM system, which is the same improvement as the DBN-DNN, while on 430-h the AE-BN system provides a 0.5% improvement over the baseline. The 17.5% WER is the best result to date on the Dev-04f task, using an acoustic model trained on 50 h of data. Finally, the complementarity of the AE-BN and baseline methods is explored by performing model combination on both the 50- and 430-h tasks. Table 4 shows that model-combination provides an additional 1.1% absolute improvement over individual systems on the 50-h task, and a 0.5% absolute improvement over the individual systems on the 430-h task, confirming the complementarity of the AE-BN and baseline systems.

[THE SUCCESSES ACHIEVED USING PRETRAINING LED TO A RESURGENCE OF INTEREST IN DNNS FOR ACOUSTIC MODELING.]

USING DNNs TO ESTIMATE ARTICULATORY FEATURES FOR DETECTION-BASED SPEECH RECOGNITION

A recent study [65] demonstrated the effectiveness of DBN-DNNs for detecting subphonetic speech attributes (also known as phonological or articulatory features [66]) in the widely used *The Wall Street Journal* speech database (5k-WSJ0). Thirteen MFCCs plus first- and second-temporal derivatives were used as the short-time spectral representation of the speech signal. The phone labels were derived from the forced alignments generated using a GMM-HMM system trained with ML, and that HMM system had 2,818 tied-state, crossword triphones, each modeled by a mixture of eight Gaussians. The attribute labels were generated by mapping phone labels to attributes, simplifying the overlapping characteristics of the articulatory features. The 22 attributes used in the recent work, as reported in [65], are a subset of the articulatory features explored in [66] and [67].

[TABLE 4] WER IN % ON ENGLISH BROADCAST NEWS.

	50 H		430 H	
LVCSR STAGE	GMM-HMM BASELINE	AE-BN	GMM/HMM BASELINE	AE-BN
FSA	24.8	20.6	20.2	17.6
+fBMMI	20.7	19.0	17.7	16.6
+BMMI	19.6	18.1	16.5	15.8
+MLLR	18.8	**17.5**	16.0	**15.5**
MODEL COMBINATION	16.4		15.0	

DBN-DNNs achieved less than half the error rate of shallow neural nets with a single hidden layer. DNN architectures with five to seven hidden layers and up to 2,048 hidden units per layer were explored, producing greater than 90% frame-level accuracy for all 21 attributes tested in the full DNN system. On the same data, DBN-DNNs also achieved a very high per frame phone classification accuracy of 86.6%. This level of accuracy for detecting subphonetic fundamental speech units may allow a new family of flexible speech recognition and understanding systems that make use of phonological features in the full detection-based framework discussed in [65].

SUMMARY AND FUTURE DIRECTIONS

When GMMs were first used for acoustic modeling, they were trained as generative models using the EM algorithm, and it was some time before researchers showed that significant gains could be achieved by a subsequent stage of discriminative training using an objective function more closely related to the ultimate goal of an ASR system [7], [68]. When neural nets were first used, they were trained discriminatively. It was only recently that researchers showed that significant gains could be achieved by adding an initial stage of generative pretraining that completely ignores the ultimate goal of the system. The pretraining is much more helpful in deep neural nets than in shallow ones, especially when limited amounts of labeled training data are available. It reduces overfitting, and it also reduces the time required for discriminative fine-tuning with backpropagation, which was one of the main impediments to using DNNs when neural networks were first used in place of GMMs in the 1990s. The successes achieved using pretraining led to a resurgence of interest in DNNs for acoustic modeling. Retrospectively, it is now clear that most of the gain comes from using DNNs to exploit information in neighboring frames and from modeling tied context-dependent states. Pretraining is helpful in reducing overfitting, and it does reduce the time taken for fine-tuning, but similar reductions in training time can be achieved with less effort by careful choice of the scales of the initial random weights in each layer.

The first method to be used for pretraining DNNs was to learn a stack of RBMs, one per hidden layer of the DNN. An RBM is an undirected generative model that uses binary latent variables, but training it by ML is expensive, so a much faster, approximate method called CD is used. This method has strong similarities to training an autoencoder network (a nonlinear version of PCA) that converts each datapoint into a code from which it is easy to approximately reconstruct the datapoint. Subsequent research showed that autoencoder networks with one layer of logistic hidden units also work well for pretraining, especially if they are regularized by adding noise to the inputs or by constraining the codes to be insensitive to small changes in the input. RBMs do not require such regularization because the Bernoulli noise introduced by using stochastic binary hidden units acts as a very strong regularizer [21].

We have described how three major speech research groups achieved significant improvements in a variety of state-of-the-art ASR systems by replacing GMMs with DNNs, and we believe that there is the potential for considerable further improvement. There is no reason to believe that we are currently using the optimal types of hidden units or the optimal network architectures, and it is highly likely that both the pretraining and fine-tuning algorithms can be modified to reduce the amount of overfitting and the amount of computation. We therefore expect that the performance gap between acoustic models that use DNNs and ones that use GMMs will continue to increase for some time.

[CURRENTLY, THE BIGGEST DISADVANTAGE OF DNNs COMPARED WITH GMMs IS THAT IT IS MUCH HARDER TO MAKE GOOD USE OF LARGE CLUSTER MACHINES TO TRAIN THEM ON MASSIVE DATA SETS.]

Currently, the biggest disadvantage of DNNs compared with GMMs is that it is much harder to make good use of large cluster machines to train them on massive data sets. This is offset by the fact that DNNs make more efficient use of data so they do not require as much data to achieve the same performance, but better ways of parallelizing the fine-tuning of DNNs is still a major issue.

AUTHORS

Geoffrey Hinton (geoffrey.hinton@gmail.com) received his Ph.D. degree from the University of Edinburgh in 1978. He spent five years as a faculty member at Carnegie Mellon University, Pittsburgh, Pennsylvania, and he is currently a distinguished professor at the University of Toronto. He is a fellow of the Royal Society and an honorary foreign member of the American Academy of Arts and Sciences. His awards include the David E. Rumelhart Prize, the International Joint Conference on Artificial Intelligence Research Excellence Award, and the Gerhard Herzberg Canada Gold Medal for Science and Engineering. He was one of the researchers who introduced the back-propagation algorithm. His other contributions include Boltzmann machines, distributed representations, time-delay neural nets, mixtures of experts, variational learning, CD learning, and DBNs.

Li Deng (deng@microsoft.com) received his Ph.D. degree from the University of Wisconsin–Madison. In 1989, he joined the Department of Electrical and Computer Engineering at the University of Waterloo, Ontario, Canada, as an assistant professor, where he became a tenured full professor in 1996. In 1999, he joined MSR, Redmond, Washington, as a senior researcher, where he is currently a principal researcher. Prior to MSR, he also worked or taught at Massachusetts Institute of Technology, ATR Interpreting Telecommunications Research Laboratories (Kyoto, Japan), and Hong Kong University of Science and Technology. In the general areas of speech recognition, signal processing, and machine learning, he has published over 300 refereed papers in leading journals and conferences and three books. He is a Fellow of the Acoustical Society of America, the International Speech Communication Association (ISCA) and the IEEE. He was ISCA's Distinguished Lecturer in 2010–2011. He has been granted over 50 patents and has received awards/honors bestowed by the IEEE, ISCA, the Acoustical Society of America (ASA), Microsoft, and other organizations including the latest 2011 IEEE Signal Processing Society (SPS) Meritorious Service Award. He served on the Board of Governors of the IEEE SPS (2008–2010), and as editor-in-chief of *IEEE Signal Processing Magazine* (2009–2011). He is currently the editor-in-chief of *IEEE Transactions on Audio, Speech, and Language Processing* (2012–2014). He is the general chair of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2013.

Dong Yu (dongyu@ieee.org) received a Ph.D. degree in computer science from the University of Idaho, an M.S. degree in computer science from Indiana University at Bloomington, an M.S. degree in electrical engineering from the Chinese Academy of Sciences, and a B.S. degree (with honors) in electrical engineering from Zhejiang University (China). He joined Microsoft Corporation in 1998 and MSR 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 90 papers in these areas and is the inventor/coinventor of more than 40 granted/pending patents. He is currently an associate editor of *IEEE Transactions on Audio, Speech, and Language Processing* (2011–present) and has been an associate editor of *IEEE Signal Processing Magazine* (2008–2011) and was the lead guest editor of the Special Issue on Deep Learning for Speech and Language Processing (2010–2011), *IEEE Transactions on Audio, Speech, and Language Processing*.

George E. Dahl (george.dahl@gmail.com) received a B.A. degree in computer science with highest honors from Swarthmore College and an M.Sc. degree from the University of Toronto, where he is currently completing a 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.

Abdel-rahman Mohamed (asamir@cs.toronto.edu) received his B.Sc. and M.Sc. degrees from the Department of Electronics and Communication Engineering, Cairo University in 2004 and

2007, respectively. In 2004, he worked in the speech research group at RDI Company, Egypt. He then joined the ESAT-PSI speech group at the Katholieke Universiteit Leuven, Belgium. In September 2008, he started his Ph.D. degree at the University of Toronto. His research focus is in developing machine learning techniques to advance human language technologies.

Navdeep Jaitly (ndjaitly@yahoo.com) received his B.A. degree from Hanover College and an M.Math degree from the University of Waterloo in 2000. After receiving his master's degree, he developed algorithms and statistical methods for analysis of protoemics data at Caprion Pharmaceuticals in Montreal and at Pacific Northwest National Labs in Washington. Since 2008, he has been pursuing a Ph.D. degree at the University of Toronto. His current interests lie in machine learning, speech recognition, computational biology, and statistical methods.

Andrew Senior (andrewsenior@google.com) received his Ph.D. degree from the University of Cambridge and is a research scientist at Google. Before joining Google, he worked at IBM Research in the areas of handwriting, audio-visual speech, face, and fingerprint recognition as well as video privacy protection and visual tracking. He edited *Privacy Protection in Video Surveillance*, coauthored Springer's *Guide to Biometrics* and over 60 scientific papers, holds 26 patents, and is an associate editor of the journal *Pattern Recognition*. His research interests range across speech and pattern recognition, computer vision, and visual art.

Vincent Vanhoucke (vanhoucke@google.com) received his Ph.D. degree from Stanford University in 2004 for research in acoustic modeling and is a graduate from the Ecole Centrale Paris. From 1999 to 2005, he was a research scientist with the speech R&D team at Nuance, in Menlo Park, California. He is currently a research scientist at Google Research, Mountain View, California, where he manages the speech quality research team. Previously, he was with Like.com (now part of Google), where he worked on object, face, and text recognition technologies.

Patrick Nguyen (drpng@google.com) received his doctorate degree from the Swiss Federal Institute for Technology (EPFL) in 2002. In 1998, he founded a company developing a platform real-time foreign exchange trading. He was with the Panasonic Speech Technology Laboratory from 2000 to 2004, in Santa Barbara, California, and MSR in Redmond, Washington, from 2004 to 2010. He is currently a research scientist at Google Research, Mountain View, California. His area of expertise revolves around statistical processing of human language, and in particular, speech recognition. He is mostly known for segmental conditional random fields and eigenvoices. He was on the organizing committee of the 2011 IEEE Workshop on Automatic Speech Recognition and Understanding and he co-led the 2010 Johns Hopkins University Workshop on Speech Recognition. He currently serves on the Speech and Language Technical Committee of the IEEE SPS.

Tara Sainath (tsainath@us.ibm.com) received her Ph.D. degree in electrical engineering and computer science from Massachusetts Institute of Technology in 2009. The main focus of her Ph.D. work was in acoustic modeling for noise robust speech recognition. She joined the Speech and Language Algorithms group at IBM T.J. Watson Research Center upon completion of her Ph.D. degree. She organized a special session on sparse representations at INTERSPEECH 2010 in Japan. In addition, she has been a staff reporter of *IEEE Speech and Language Processing Technical Committee Newsletter*. She currently holds 15 U.S. patents. Her research interests mainly focus in acoustic modeling, including sparse representations, DBN works, adaptation methods, and noise robust speech recognition.

Brian Kingsbury (bedk@us.ibm.com) received the B.S. degree (high honors) in electrical engineering from Michigan State University, East Lansing, in 1989 and the Ph.D. degree in computer science from the University of California, Berkeley, in 1998. Since 1999, he has been a research staff member in the Department of Human Language Technologies, IBM T.J. Watson Research Center, Yorktown Heights, New York. His research interests include large-vocabulary speech transcription, audio indexing and analytics, and information retrieval from speech. From 2009 to 2011, he served on the IEEE SPS's Speech and Language Technical Committee, and from 2010 to 2012 he was an ICASSP area chair. He is currently an associate editor of *IEEE Transactions on Audio, Speech, and Language Processing*.

REFERENCES

[1] J. Baker, L. Deng, J. Glass, S. Khudanpur, Chin Hui Lee, N. Morgan, and D. O'Shaughnessy, "Developments and directions in speech recognition and understanding, part 1," *IEEE Signal Processing Mag.*, vol. 26, no. 3, pp. 75–80, May 2009.

[2] S. Furui, Digital Speech Processing, Synthesis, and Recognition. New York: Marcel Dekker, 2000.

[3] B. H. Juang, S. Levinson, and M. Sondhi, "Maximum likelihood estimation for multivariate mixture observations of Markov chains," *IEEE Trans. Inform. Theory*, vol. 32, no. 2, pp. 307–309, 1986.

[4] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *J. Acoust. Soc. Amer.*, vol. 87, no. 4, pp. 1738–1752, 1990.

[5] S. Furui, "Cepstral analysis technique for automatic speaker verification," *IEEE Trans. Acoust., Speech, Signal, Processing*, vol. 29, no. 2, pp. 254–272, 1981.

[6] S. Young, "Large vocabulary continuous speech recognition: A review," *IEEE Signal Processing Mag.*, vol. 13, no. 5, pp. 45–57, 1996.

[7] L. Bahl, P. Brown, P. de Souza, and R. Mercer, "Maximum mutual information estimation of hidden Markov model parameters for speech recognition," in *Proc. ICASSP*, 1986, pp. 49–52.

[8] H. Hermansky, D. P. W. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional HMM systems," in *Proc. ICASSP*. Los Alamitos, CA: IEEE Computer Society, 2000, vol. 3, pp. 1635–1638.

[9] H. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*, Norwell, MA: Kluwer, 1993.

[10] L. Deng, "Computational models for speech production," in *Computational Models of Speech Pattern Processing,* K. M. Ponting, Ed. New York: Springer-Verlag, 1999, pp. 199–213.

[11] L. Deng, "Switching dynamic system models for speech articulation and acoustics," in *Mathematical Foundations of Speech and Language Processing*, M. Johnsonm S. P. Khudanpur, M. Ostendorf, and R. Rosenfeld. New York: Springer-Verlag, 2003, pp. 115–134.

[12] A. Mohamed, G. Dahl, and G. Hinton, "Deep belief networks for phone recognition," in *Proc. NIPS Workshop Deep Learning for Speech Recognition and Related Applications*, 2009.

[13] A. Mohamed, G. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. Audio Speech Lang. Processing*, vol. 20, no. 1, pp. 14–22, Jan. 2012.

[14] 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.

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

[16] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural Comput.*, vol. 22, no. 12, pp. 3207–3220, 2010.

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

[18] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," in *Proc. 24th Int. Conf. Machine Learning*, 2007, pp. 473–480.

[19] J. Pearl, Probabilistic Inference in Intelligent Systems: Networks of Plausible Inference. San Mateo, CA: Morgan Kaufmann, 1988.

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

[21] G. E. Hinton, "A practical guide to training restricted Boltzmann machines," Tech. Rep. UTML TR 2010-003, Dept. Comput. Sci., Univ. Toronto, 2010.

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

[23] T. N. Sainath, B. Ramabhadran, and M. Picheny, "An exploration of large vocabulary tools for small vocabulary phonetic recognition," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, 2009, pp. 359–364.

[24] A. Mohamed, T. N. Sainath, G. E. Dahl, B. Ramabhadran, G. E. Hinton, and M. Picheny, "Deep belief networks using discriminative features for phone recognition," in *Proc. ICASSP*, 2011, pp. 5060–5063.

[25] A. Mohamed, G. Hinton, and G. Penn, "Understanding how deep belief networks perform acoustic modelling," in *Proc. ICASSP*, 2012, pp. 4273–4276.

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

[27] A. Robinson, "An application to recurrent nets to phone probability estimation," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 298–305, 1994.

[28] J. Ming and F. J. Smith, "Improved phone recognition using Bayesian triphone models," in *Proc. ICASSP*, 1998, pp. 409–412.

[29] L. Deng and D. Yu, "Use of differential cepstra as acoustic features in hidden trajectory modelling for phonetic recognition," in *Proc. ICASSP*, 2007, pp. 445–448.

[30] A. Halberstadt and J. Glass, "Heterogeneous measurements and multiple classifiers for speech recognition," in *Proc. ICSLP*, 1998.

[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] T. N. Sainath, B. Ramabhadran, M. Picheny, D. Nahamoo, and D. Kanevsky, "Exemplar-based sparse representation features: From TIMIT to LVCSR," *IEEE Trans. Audio Speech Lang. Processing*, vol. 19, no. 8, pp. 2598–2613, Nov. 2011.

[33] G. E. Dahl, M. Ranzato, A. Mohamed, and G. E. Hinton, "Phone recognition with the mean-covariance restricted Boltzmann machine," in *Advances in Neural Information Processing Systems 23*, J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, Eds. Cambridge, MA: MIT Press, 2010, pp. 469–477.

[34] O. Abdel-Hamid, A. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition," in *Proc. ICASSP*, 2012, pp. 4277–4280.

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

[36] Y. Bengio, R. De Mori, G. Flammia, and F. Kompe, "Global optimization of a neural network—Hidden Markov model hybrid," in *Proc. EuroSpeech*, 1991.

[37] B. Kingsbury, "Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling," in *Proc. ICASSP*, 2009, pp. 3761–3764.

[38] R. Prabhavalkar and E. Fosler-Lussier, "Backpropagation training for multilayer conditional random field based phone recognition," in *Proc. ICASSP*, 2010, pp. 5534–5537.

[39] H. Lee, P. Pham, Y. Largman, and A. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Cambridge, MA: MIT Press, 2009, pp. 1096–1104.

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

[41] H. Zen, M. Gales, Y. Nankaku, and K. Tokuda, "Product of experts for statistical parametric speech synthesis," *IEEE Trans. Audio Speech and Lang. Processing*, vol. 20, no. 3, pp. 794–805, Mar. 2012.

[42] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pretrained deep neural networks for large-vocabulary speech recognition," *IEEE Trans. Audio Speech Lang. Processing*, vol. 20, no. 1, pp. 30–42, Jan. 2012.

[43] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proc. Interspeech*, 2011, pp. 437–440.

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

[45] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. IEEE ASRU*, 2011, pp. 24–29.

[46] 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.

[47] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "An application of pretrained deep neural networks to large vocabulary speech recognition," submitted for publication.

[48] G. Zweig, P. Nguyen, D. V. Compernolle, K. Demuynck, L. Atlas, P. Clark, G. Sell, M. Wang, F. Sha, H. Hermansky, D. Karakos, A. Jansen, S. Thomas, G. S. V. S. Sivaram, S. Bowman, and J. Kao, "Speech recognition with segmental conditional random fields: A summary of the JHU CLSP 2010 summer workshop," in *Proc. ICASSP*, 2011, pp. 5044–5047.

[49] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," in *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*, 2011 [Online]. Available: http://research.google.com/pubs/archive/37631.pdf

[50] T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Improvements in using deep belief networks for large vocabulary continuous speech recognition," Speech and Language Algorithm Group, IBM, Yorktown Heights, NY, Tech. Rep. UTML TR 2010-003, Feb. 2011.

[51] L. Deng and D. Yu, "Deep convex network: A scalable architecture for speech pattern classification," in *Proc. Interspeech*, 2011, pp. 2285–2288.

[52] L. Deng, D. Yu, and J. Platt, "Scalable stacking and learning for building deep architectures," in *Proc. ICASSP*, 2012, pp. 2133–2136.

[53] D. Yu, L. Deng, G. Li, and Seide F, "Discriminative pretraining of deep neural networks," U.S. Patent Filing, Nov. 2011.

[54] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 11, pp. 3371–3408, 2010.

[55] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive autoencoders: Explicit invariance during feature extraction," in *Proc. 28th Int. Conf. Machine Learning*, 2011, pp. 833–840.

[56] C. Plahl, T. N. Sainath, B. Ramabhadran, and D. Nahamoo, "Improved pretraining of deep belief networks using sparse encoding symmetric machines," in *Proc. ICASSP*, 2012, pp. 4165–4168.

[57] B. Hutchinson, L. Deng, and D. Yu, "A deep architecture with bilinear modeling of hidden representations: Applications to phonetic recognition," in *Proc. ICASSP*, 2012, pp. 4805–4808.

[58] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *Proc. 28th Int. Conf. Machine Learning*, 2011, pp. 265–272.

[59] J. Martens, "Deep learning via Hessian-free optimization," in *Proc. 27th Int. Conf. Machine learning*, 2010, pp. 735–742.

[60] N. Morgan, "Deep and wide: Multiple layers in automatic speech recognition," *IEEE Trans. Audio Speech Lang. Processing*, vol. 20, no. 1, Jan. 2012, pp. 7–13.

[61] G. Sivaram and H. Hermansky, "Sparse multilayer perceptron for phoneme recognition," *IEEE Trans. Audio Speech Lang. Processing*, vol. 20, no. 1, Jan. 2012, pp. 23–29.

[62] T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Auto-encoder bottleneck features using deep belief networks," in *Proc. ICASSP*, 2012, pp. 4153–4156.

[63] 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. Cretin, H. Bourlard, and M. Athineos, "Pushing the envelope aside," *IEEE Signal Processing Mag.*, vol. 22, no. 5, pp. 81–88, Sept. 2005.

[64] O. Vinyals and S. V. Ravuri, "Comparing multilayer perceptron to deep belief network tandem features for robust ASR," in *Proc. ICASSP*, 2011, pp. 4596–4599.

[65] D. Yu, S. Siniscalchi, L. Deng, and C. Lee, "Boosting attribute and phone estimation accuracies with deep neural networks for detection-based speech recognition," in *Proc. ICASSP*, 2012, pp. 4169–4172.

[66] L. Deng and D. Sun, "A statistical approach to automatic speech recognition using the atomic speech units constructed from overlapping articulatory features," *J. Acoust. Soc. Amer.*, vol. 85, no. 5, pp. 2702–2719, 1994.

[67] J. Sun and L. Deng, "An overlapping-feature based phonological model incorporating linguistic constraints: Applications to speech recognition," *J. Acoustic. Soc. Amer.*, vol. 111, no. 2, pp. 1086–1101, 2002.

[68] P. C. Woodland and D. Povey, "Large scale discriminative training of hidden Markov models for speech recognition," *Comput Speech Lang.*, vol. 16, no. 1, pp. 25–47, 2002.

[69] F. Grezl, M. Karaat, S. Kontar, and J. Cernocky. "Probabilistic and bottle-neck features for LVCSR of meetings," in *Proc. ICASSP*, 2007.