Deep Belief Networks for phone recognition

Abdel-rahman Mohamed, George Dahl, and Geoffrey Hinton
Department of Computer Science
University of Toronto
{asamir,gdahl,hinton}@cs.toronto.edu

Abstract

Hidden Markov Models (HMMs) have been the state-of-the-art techniques for acoustic modeling despite their unrealistic independence assumptions and the very limited representational capacity of their hidden states. There are many proposals in the research community for deeper models that are capable of modeling the many types of variability present in the speech generation process. Deep Belief Networks (DBNs) have recently proved to be very effective for a variety of machine learning problems and this paper applies DBNs to acoustic modeling. On the standard TIMIT corpus, DBNs consistently outperform other techniques and the best DBN achieves a phone error rate (PER) of 23.0% on the TIMIT core test set.

1 Introduction

A state-of-the-art Automatic Speech Recognition (ASR) system typically uses Hidden Markov Models (HMMs) to model the sequential structure of speech signals, with local spectral variability modeled using mixtures of Gaussian densities. HMMs make two main assumptions. The first assumption is that the hidden state sequence can be well-approximated using a first order Markov chain where each state S_t at time t depends only on S_{t-1}. Second, observations at different time steps are assumed to be conditionally independent given a state sequence. Although these assumptions are not realistic, they enable tractable decoding and learning even with large amounts of speech data. Many methods have been proposed for relaxing the very strong conditional independence assumptions of standard HMMs (e.g. [1, 2, 3, 4]). Substantial research effort has been devoted to going beyond the "beads-on-a-string" view of speech to representing structure in speech above the level of the phonetic segment [5, 6, 7]. This has led to promising results using segment- and landmark-based methods for phone recognition (e.g, [8, 9]).

The limited representational capacity of HMMs prevents them from modeling streams of interacting knowledge sources in the speech signal which may require deeper architectures with multiple layers of representations. The work in [10] proposes a hierarchical framework where each layer is designed to capture a set of distinctive feature landmarks. For each feature, a specialized acoustic representation is constructed in which that feature best expresses itself. In [11], a probabilistic generative model is introduced where the dynamic structure in the hidden vocal tract resonance space is used to characterize long-span contextual influence across phonetic units. Feedforward neural networks were used in other multilayer frameworks, such as the TRAP architecture [12]. The TRAP architecture uses a one second long feature vector that describes segments of temporal evolution of critical-band spectral densities within a single critical band. Sub-word posterior probabilities are estimated using feedforward neural networks for each critical band which are merged to produce the final estimation of posterior probabilities using another feedforward neural network in the last layer. In [13], the split temporal context system is introduced which modifies the TRAP system by including splits over time as well as over frequency bands in the middle layer of the system before the final merger neural network.

In this work, we propose using Deep Belief Networks (DBNs) [14] to model the spectral variabilities in speech. DBNs are probabilistic generative models that are composed of multiple layers of stochastic latent variables with Restricted Boltzmann Machines (RBMs) as their building blocks. DBNs have a greedy layer-wise unsupervised learning algorithm as well as a discriminative fine-tuning procedure for optimizing performance on classification tasks.

DBNs and related models have been used successfully for hand-written character recognition [14, 15], 3-D object recognition [16], information retrieval [17, 18], motion capture data modeling [19, 20], and machine transliteration [21].

2 Deep belief networks

Learning is difficult in densely connected, directed belief nets that have many hidden layers because it is difficult to infer the posterior distribution over the hidden variables, when given a data vector, due to the phenomenon of explaining away. Markov chain Monte Carlo methods [22] can be used to sample from the posterior, but they are typically very time-consuming.

In [14] complementary priors were used to eliminate the explaining away effects producing a training procedure which is equivalent to training a sequence of restricted Boltzmann machines (RBMs) [23]. An RBM is a bipartite graph in which visible units that represent observations are connected to binary, stochastic hidden units using undirected weighted connections. They are restricted in the sense that there are no visible-visible or hidden-hidden connections. RBMs have an efficient training procedure which makes them suitable as building blocks for Deep Belief Networks (DBNs).

2.1 Restricted Boltzmann machines

An RBM [figure 1-(a)] is a particular type of Markov Random Field (MRF) that has one layer of binary stochastic hidden units and one layer of binary stochastic visible units, although the units need not be Bernoulli random variables and can in fact have any distribution in the exponential family [24]. Typically, all visible units are connected to all hidden units. The weights on the connections and the biases of the individual units define a probability distribution over the binary state vectors $\mathbf{v}$ of the visible units via an energy function. The energy of the joint configuration $(\mathbf{v}, \mathbf{h})$ is given by [24]:

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\sum_{i=1}^{\mathcal{V}} \sum_{j=1}^{\mathcal{H}} w_{ij} v_i h_j - \sum_{i=1}^{\mathcal{V}} b_i v_i - \sum_{j=1}^{\mathcal{H}} a_j h_j \tag{1}$$

where $\theta = (\mathbf{w}, \mathbf{b}, \mathbf{a})$ and w_{ij} represents the symmetric interaction term between visible unit i and hidden unit j while b_i and a_j are their bias terms. $\mathcal{V}$ and $\mathcal{H}$ are the numbers of visible and hidden units. The probability that the model assigns to a visible vector $\mathbf{v}$ is:

$$p(\mathbf{v}; \theta) = \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{u}} \sum_{\mathbf{h}} e^{-E(\mathbf{u},\mathbf{h})}} \tag{2}$$

Since there are no hidden-hidden or visible-visible connections, the conditional distributions $p(\mathbf{v}|\mathbf{h})$ and $p(\mathbf{h}|\mathbf{v})$ are factorial and are given by:

$$\begin{aligned} p(h_j = 1|\mathbf{v}; \theta) &= \sigma(\sum_{i=1}^{\mathcal{V}} w_{ij} v_i + a_j) \\ p(v_i = 1|\mathbf{h}; \theta) &= \sigma(\sum_{j=1}^{\mathcal{H}} w_{ij} h_j + b_i), \end{aligned} \tag{3}$$

where $\sigma(x) = (1 + e^{-x})^{-1}$. To train an RBM to model the joint distribution of data and class labels, the visible vector is concatenated with a binary vector of class labels. The energy function becomes:

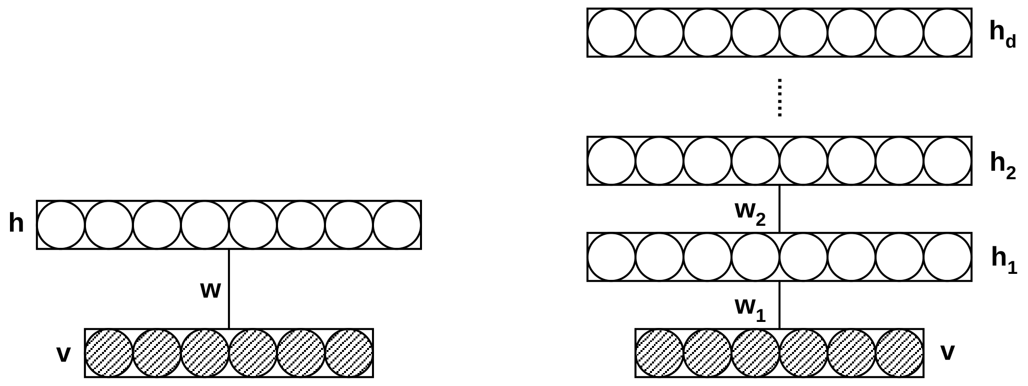
$$E(\mathbf{v},\mathbf{l},\mathbf{h};\theta) = -\sum_{i=1}^{\mathcal{V}}\sum_{j=1}^{\mathcal{H}} w_{ij}h_jv_i - \sum_{y=1}^{\mathcal{L}}\sum_{j=1}^{\mathcal{H}} w_{yj}h_jl_y - \sum_{j=1}^{\mathcal{H}} a_jh_j - \sum_{y=1}^{\mathcal{L}} c_yl_y - \sum_{i=1}^{\mathcal{V}} b_iv_i \quad (4)$$

$$p(l_y = 1|\mathbf{h};\theta) = \text{softmax}(\sum_{j=1}^{\mathcal{H}} w_{yj}h_j + c_y). \quad (5)$$

Furthermore, $p(\mathbf{l}|\mathbf{v})$ can be computed exactly using:

$$p(\mathbf{l}|\mathbf{v}) = \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{l},\mathbf{h})}}{\sum_{\mathbf{l}}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{l},\mathbf{h})}}. \quad (6)$$

The value of $p(\mathbf{l}|\mathbf{v})$ can be computed efficiently by exploiting the conditional independence of the hidden units, which allows the hidden units to be marginalized out in a time that is linear in the number of hidden units.



(a) RBM (b) DBN

Figure 1: The DBN is composed of RBMs.

2.2 RBM training

2.2.1 Generative training of an RBM

Following the gradient of the joint likelihood function of data and labels, the update rule for the visible-hidden weights is

$$\Delta w_{ij} = \langle v_ih_j\rangle_{data} - \langle v_ih_j\rangle_{model} \quad (7)$$

The expectation $\langle v_ih_j\rangle_{data}$ is the frequency with which the visible unit v_i and the hidden unit h_j are on together in the training set and $\langle v_ih_j\rangle_{model}$ is that same expectation under the distribution defined by the model. The term $\langle . \rangle_{model}$ takes exponential time to compute exactly so the Contrastive Divergence (CD) approximation to the gradient is used instead [15]. The new update rule becomes:

$$\Delta w_{ij} = \langle v_ih_j\rangle_{data} - \langle v_ih_j\rangle_1 \quad (8)$$

where $\langle . \rangle_1$ represents the expectation with respect to the distribution of samples from running the Gibbs sampler initialized at the data for one full step.

2.2.2 Discriminative training of a DBN using backpropagation

The RBM pretraining procedure of a DBN can be used to initialize the weights of a deep neural network, which can then be discriminatively fine-tuned by backpropagating error derivatives. The "recognition" weights of the DBN become the weights of a standard neural network.

2.2.3 Hybrid training of an RBM

In cases where the RBM models the joint distribution of visible data and class labels, a hybrid training procedure can be used to fine-tune the generatively trained parameters. Since the log conditional probability, $log\ p(\mathbf{l}|\mathbf{v})$, can be computed exactly, the gradient can also be computed exactly. The update rule for the visible-hidden weights is

$$\Delta w_{ij} = \sum_{j=1}^{\mathcal{H}} \sigma(a_j + w_{jy} + \sum_{i=1}^{\mathcal{V}} w_{ij} v_i) v_i - \sum_{l=1}^{\mathcal{L}} \sum_{j=1}^{\mathcal{H}} \sigma(a_j + w_{jl} + \sum_{i=1}^{\mathcal{V}} w_{ij} v_i) p(l|\mathbf{v}) v_i \tag{9}$$

To avoid model overfitting, we follow the gradient of a hybrid function $f(\mathbf{v}, \mathbf{l})$ which contains both generative and discriminative components:

$$f(\mathbf{v}, \mathbf{l}) = \alpha p(\mathbf{l}|\mathbf{v}) + p(\mathbf{v}|\mathbf{l}) \tag{10}$$

In this case $p(\mathbf{v}|\mathbf{l})$ works as a regularizer and is learned by using the original labels with the reconstructed data to infer the states of the hidden units at the end of the sampling step. The α parameter controls the emphasis given to the discriminative component of the objective function. Since the original labels are used during hidden layer reconstruction for evaluating $p(\mathbf{v}|\mathbf{l})$, the label biases are updated using the gradient of $p(\mathbf{l}|\mathbf{v})$ only.

2.3 DBN structure

Each layer of hidden units learns to represent features that capture higher order correlations in the original input data [figure 1-(b)]. The key idea behind training a deep belief net by training a sequence of RBMs is that the model parameters, θ, learned by an RBM define both $p(\mathbf{v}|\mathbf{h}, \theta)$ and the prior distribution over hidden vectors, $p(\mathbf{h}|\theta)$, so the probability of generating a visible vector, $\mathbf{v}$, can be written as:

$$p(\mathbf{v}) = \sum_{\mathbf{h}} p(\mathbf{h}|\theta) p(\mathbf{v}|\mathbf{h}, \theta) \tag{11}$$

After learning θ, $p(\mathbf{v}|\mathbf{h}, \theta)$ is kept while $p(\mathbf{h}|\theta)$ can be replaced by a better model that is learned by treating the hidden activity vectors produced from the training data as the training data for another RBM. This replacement improves a variational lower bound on the probability of the training data under the composite model [14]. So a DBN can be viewed as an RBM that defines a prior over the top layer of hidden variables in a directed belief net, combined with a set of "recognition" weights to perform fast approximate inference.

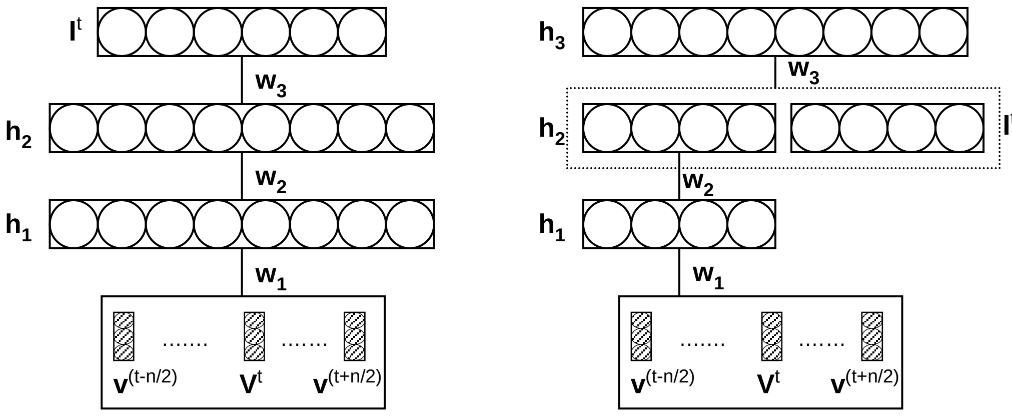
2.4 Using DBNs for phone recognition

In order to apply DBNs with fixed input and output dimensionality to phone recognition, a context window of n successive frames of feature vectors is used to set the states of the visible units of the lower layer of the DBN which produces a probability distribution over the possible labels of the the central frame. To generate phone sequences, a sequence of probability distributions over the possible labels for each frame are fed into a standard Viterbi decoder.

We employed two general types of DBN architectures. Both types use greedy layer-wise Contrastive Divergence (CD) pretraining for initializing weights. The first architecture [figure 2-(a)] adds a final layer of variables that represent the desired outputs then performs a purely discriminative fine-tuning phase using backpropagation. We refer to this architecture as "BP-DBN." The second type used an RBM associative memory for the final layer to model the joint density of the labels and inputs [figure 2-(b)]. For fine-tuning, derivatives of the hybrid objective function in 10 are followed. Only the discriminative component of the weight updates is propagated back through the earlier layers in the network during the fine-tuning stage. We refer to this architecture as "AM-DBN."

2.5 Generalized softmax (GSM) output layer

When the number of possible classes is very large and the distribution of frequencies for different classes is far from uniform, it may sometimes be advantageous to use a different encoding for



(a)A 2-layer BP-DBN (b)A 3-layer AM-DBN

Figure 2: The DBN architectures used in this work.

the class targets than the standard one-of-K softmax encoding [25]. It is quite straightforward to use an arbitrary fixed binary code for each class. Suppose we represent each class with its own q-dimensional code vector and thus we have q output units for our model. Let z be the q-dimensional column vector generated by the network for a certain input speech segment; z needs to be transformed into a vector of class posterior probabilities. If C_j is the row vector holding the code for class j, then the expression for the probability the model assigns to class t given z becomes

$$P(t|\theta, z) = \frac{e^{C_t z}}{\sum_j e^{C_j z}} \tag{12}$$

If we allow C_j to be the jth row of the identity matrix, we recover the normal softmax expression.

3 Experimental setup

3.1 TIMIT corpus

Phone recognition experiments were performed on the TIMIT corpus [1]. The 462 speaker training set was used. All SA records (i.e., identical sentences for all speakers in the database) were removed as they could bias the results. A development set of 50 speakers was used for model tuning. Results are reported using the 24-speaker core test set. The speech was analyzed using a 25-ms Hamming window with 10-ms between the left edges of successive frames. In all the experiments, we represented the speech using 12th-order Mel frequency cepstral coefficients (MFCCs) and energy, along with their first and second temporal derivatives. The data were normalized to have zero mean and unit variance over the entire corpus. All experiments used a context window of 11 frames as the visible states. We used 183 target class labels (i.e., 3 states for each one of the 61 phones). After decoding, starting and ending silences were removed and the 61 phone classes were mapped to a set of 39 classes as in [26] for scoring. All of our experiments used a bigram language model over phones, estimated from the training set. The decoder parameters were tuned to optimize performance on the development set for each run using grid search.

3.2 Computational setup

Training DBNs of the sizes used in this paper is quite computationally expensive. Training was accelerated by exploiting a graphics processor. A single pass over the entire training set during pretraining took about 5 minutes. An epoch of fine-tuning with backpropagation took around 13 minutes. The discriminative gradient computation for hybrid training was substantially more expensive. Each epoch of hybrid fine-tuning took around an hour. These time estimates represent the

[1]http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1.

largest architecture running on one of the GPUs in a NVIDIA Tesla S1070 system, using the library in [27].

4 Experiments

For all experiments, the Viterbi decoder parameters (i.e. the word insertion probability, the language model scale factor) were optimized on the development set and then the best performing setting was used to compute the phone error rate (PER) for the core test set.

Figure 3 explores the effect of varying the number of hidden layers in the model. The BP-DBN architecture is used with 2048 hidden units per layer. All models use the same fixed random binary code matrix to convert the 128 network output units into probabilities over the target 183 states' labels.

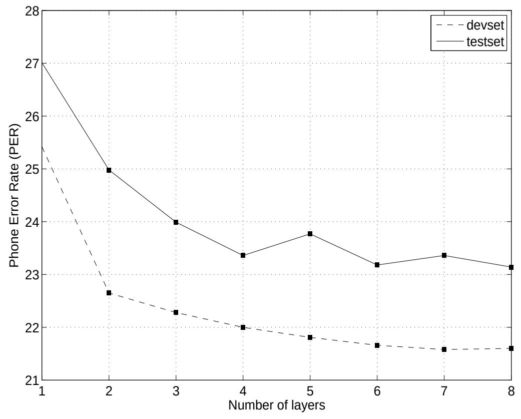


Figure 3: The effect of the model depth on PER.

Adding a second layer significantly reduces the PER as shown in figure 3. By adding more layers, the PER on the development set starts to plateau while the PER on the test stays between 23% and 24% after adding the 4^{th} layer. This motivates the decision to restrict most of the experiments to four and five layer models.

Phone error rates (PER) for four-layer architectures with different hidden layer sizes are presented in table 1.

Table 1: *The effect of layer size on PER*

Model	devset	testset
1024 units	21.94%	23.46%
2048 units	22.00%	23.36%
3072 units	21.74%	23.54%

We generally focused our computational resources on experiments with 2048 units per hidden layer since the performance was not significantly different for different numbers of units per layer.

To check the effect of using the Generalized softmax (GSM) in the output layer of the network, we compared its performance to the standard 183-way softmax output layer. Both a 4 hidden layer model using a 128-dimensional GSM and the same architecture using a standard softmax achieved 22% PER on the development set. On the core test set, The PER of the GSM model is 23.36% while the standard Softmax PER is 23.9%. The 128-dimensional GSM model can be viewed as a 5 layer DBN with a final layer of fixed weights. To be clear on the main source of improvement, we compared the 128-dimensional GSM model to a 5 layer DBN with a final layer of 128 hidden units and a 5 layer DBN with 2048 units in each layer. Table 2 shows that having a "bottleneck" at the last layer of the DBN is useful to combat overfitting and to encourage the model to share features between different classes. Since all but the output layer weights in a BP-DBN-style architecture are pretrained, a learned bottleneck substantially reduces the number of parameters that do not receive the benefits of pretraining. If we simply use a random fixed binary code for classes, although the model still has almost as many weights that aren't pretrained as an architecture without a bottleneck, the model is forced to share features between classes because each output code bit is on for a random subset of approximately half of the classes. Either making the last hidden layer substantially smaller than the other layers or using a fixed (not learned) transformation between the DBN output units and the actual 183 classes was sufficient for achieving good results, even with minimal weight decay. Additionally, we were unable to achieve as strong results in preliminary experiments using L2 weight decay alone. Unsurprisingly, learning the weights between the classes and the penultimate 128 unit layer produced better results than using a fixed random binary matrix; however, using an unlearned random transformation did not substantially degrade performance.

Table 2: *The effect of GSM on PER*

Model	devset	testset
128-dimensional GSM	22.00%	23.36%
4 layers + 128 hidden units	22.00%	**23.00**%
4 layers + 2048 hidden units	21.98%	23.73%

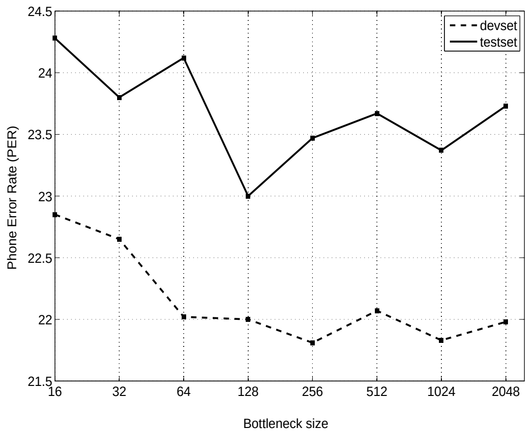


Figure 4: The effect of bottleneck size on PER.

As shown in figure 4, the quality of the results was quite resilient to changes in the size of the final hidden layer. In general, as long as there were at least 64 hidden units in the fifth hidden layer, phone error rates on the development set were practically indistinguishable. By examining results on the core test set, although many of the differences are not statistically significant, we note that some reduction in the size of fifth layer compared to the fourth layer seems helpful.

Table 3 presents the PER achieved by using the AM-DBN architecture for 2048 and 3072 Associative Memory (AM) sizes on top of a 3 layer network with 2048 units for each layer. The AM-DBN architecture, although not better than the BP-DBN architecture, has a nice mechanism for avoiding overfitting without using a bottleneck which is important in much deeper belief networks.

Table 4 compares the best performing DBN model with previously reported results on the TIMIT core test set [2]. The lowest PER on the core test set was observed using a 4 layer BP-DBN architecture of 2048 hidden units per layer plus a bottleneck layer of 128 units before the 183-way softmax which is a 1.4% absolute improvement in PER over the best reported method in the literature.

[2]In [13] a PER of 21.48% is reported on the **complete** test set of TIMIT. The speech units used are not the same as the standard TIMIT definitions.

Table 3: *PERs using AM-DBN architecture*

Model	devset	testset
3k_AM	22.39	23.85
2k_AM	22.52	23.96

Table 4: *Reported results on TIMIT core test set*

Method	PER
Stochastic Segmental Models [28]	36%
Conditional Random Field [29]	34.8%
Large-Margin GMM [30]	33%
CD-HMM [4]	27.3%
Augmented conditional Random Fields [4]	26.6%
Recurrent Neural Nets [31]	26.1%
Bayesian Triphone HMM [32]	25.6%
Monophone HTMs [33]	24.8%
Heterogeneous Classifiers [34]	24.40%
Deep Belief Networks(DBNs) (this work)	**23.0%**

5 Conclusions and future work

In this work, two types of Deep Belief Network were investigated for acoustic modeling; the back-propagation DBN (BP-DBN) and the associative memory DBN (AM-DBN) architectures. The effect of model depth and hidden layer size were investigated. Both architectures have mechanisms to avoid overfitting. The use of a “bottleneck” in the last layer of the BP-DBN proved to help avoid overfitting while hybrid generative and discriminative training prevent overfitting in the AM-DBN. Both types of DBN beat other reported results on the TIMIT core test set for a wide variety of choices of the number of hidden layers and the number of units per layer.

References

[1] Li Deng, “A generalized hidden markov model with state-conditioned trend functions of time for the speech signal,” *Signal Processing*, vol. 27, no. 1, pp. 65–78, 1992.

[2] L. Deng, “A stochastic model of speech incorporating hierarchical nonstationarity.,” *IEEE Transactions on Speech and Audio Processing*, vol. 1, no. 4, pp. 471–475, 1993.

[3] J. Bilmes, “Buried Markov models for speech recognition,” in *Proc. ICASSP*, 1999.

[4] Hifny Y. and Renals S., “Speech recognition using augmented conditional random fields,” *IEEE Transactions on Audio, Speech & Language Processing*, vol. 17, no. 2, pp. 354–365, 2009.

[5] L. Deng and K. Erler, “Structural design of a hidden markov model speech recognizer using multi-valued phonetic features: Comparison with segmental speech units,” *Journal of the Acoustical Society of America*, vol. 92, no. 6, pp. 3058–3067, 1992.

[6] M. Ostendorf, “Moving beyond the ’beads-on-a-string’ model of speech,” in *IEEE ASRU Workshop*, 1999, vol. 1, pp. 79–83.

[7] J. Sun and L. Deng, “An overlapping-feature based phonological model incorporating linguistic constraints: Applications to speech recognition,” *Journal of the Acoustical Society of America*, vol. 111, no. 2, pp. 1086–1101, 2002.

[8] James R. Glass, “A probabilistic framework for segment-based speech recognition.,” *Computer Speech & Language*, vol. 17, no. 2-3, pp. 137–152, 2003.

[9] Alexander Gutkin and Simon King, “Structural representation of speech for phonetic classification,” in *Proceedings of the 17th International Conference on Pattern Recognition (ICPR’04)*, 2004, pp. 438–441.

[10] A. Jansen and P. Niyogi, “A hierarchical point process model for speech recognition,” in *Proc.*

IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2008, pp. 4093–4096.

[11] Li Deng, Dong Yu, and A. Acero, "Structured speech modeling.," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 14, no. 5, pp. 1492–1504, 2006.

[12] Hynek Hermansky and Sangita Sharma, "Traps - classifiers of temporal patterns," in *Proc. International Conference on Spoken Language Processing (ICSLP)*, 1998, pp. 1003–1006.

[13] Petr Schwarz, Pavel Matjka, and Jan ernock, "Hierarchical structures of neural networks for phoneme recognition," in *Proc. ICASSP*, 2006, pp. 325–328.

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

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

[16] V. Nair and G. Hinton, "3-d object recognition with deep belief nets.," in *To appear in Advances in Neural Information Processing Systems 22*, 2009.

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

[18] Ruslan Salakhutdinov and Geoffrey E. Hinton, "Semantic hashing.," *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[19] G. W. Taylor, G. E. Hinton, and S. Roweis, "Modeling human motion using binary latent variables," in *Proc. NIPS*, 2007.

[20] G. W. Taylor and G. E. Hinton, "Factored conditional restricted boltzmann machines for modeling motion style," in *Proc. ICML*, 2009.

[21] T Deselaers, S. Hasan, O. Bender, and H. Ney, "A deep learning approach to machine transliteration," in *Proc. EACL Workshop on Statistical Machine Translation*, 2009, pp. 233–241.

[22] Radford M. Neal, "Connectionist learning of belief networks," *Artificial Intelligence*, vol. 56, no. 1, pp. 71–113, 1992.

[23] P. Smolensky, "Information processing in dynamical systems: foundations of harmony theory," pp. 194–281, 1986.

[24] M. Welling, M. Rosen-Zvi, and G. E. Hinton, "Exponential family harmoniums with an application to information retrieval," in *Proc. NIPS*, 2005.

[25] Thomas G. Dietterich and Ghulum Bakiri, "Solving multiclass learning problems via error-correcting output codes," *Journal of Artificial Intelligence Research*, vol. 2, pp. 263–286, 1995.

[26] K. F. Lee and H. W. Hon, "Speaker-independent phone recognition using hidden markov models," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 37, no. 11, pp. 16411648, 1989.

[27] Volodymyr Mnih, "Cudamat: a CUDA-based matrix class for python," Tech. Rep. UTML TR 2009-004, Department of Computer Science, University of Toronto, November 2009.

[28] V. V. Digalakis, M. Ostendorf, and J. R. Rohlicek, "Fast algorithms for phone classification and recognition using segment-based models," *IEEE Transactions on Signal Processing*, vol. 40, pp. 2885–2896, 1992.

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

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

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

[32] J. Ming and F. J. Smith, "Improved phone recognition using bayesian triphone models," in *Proc. ICASSP*, 1998, p. 409412.

[33] 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.

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