SEQUENCE CLASSIFICATION USING THE HIGH-LEVEL FEATURES EXTRACTED FROM DEEP NEURAL NETWORKS

Li Deng[1] and Jianshu Chen[2]
[1]Microsoft Research, One Microsoft Way, Redmond, WA, USA
[2]University of California, Los Angeles, CA, USA
deng@microsoft.com; cjs09@ucla.edu

ABSTRACT

The recent success of deep neural networks (DNNs) in speech recognition can be attributed largely to their ability to extract a specific form of high-level features from raw acoustic data for subsequent sequence classification or recognition tasks. Among the many possible forms of DNN features, what forms are more useful than others and how effective these DNN features are in connection with the different types of downstream sequence recognizers remained unexplored and are the focus of this paper. We report our recent work on the construction of a diverse set of DNN features, including the vectors extracted from the output layer and from various hidden layers in the DNN. We then apply these features as the inputs to four types of classifiers to carry out the identical sequence classification task of phone recognition. The experimental results show that the features derived from the top hidden layer of the DNN perform the best for all four classifiers, especially for the autoregressive-moving-average (ARMA) version of a recurrent neural network. The feature vector derived from the DNN's output layer performs slightly worse but better than any of the hidden layers in the DNN except the top one.

Index Terms— deep neural net, feature extraction, ARMA recurrent neural net, phone recognition

1. INTRODUCTION

The use of fully-connected feed-forward deep neural networks (DNNs) in the architecture of DNN-HMM (hidden Markov Model) has dramatically reduced speech recognition errors in recent years [4][5][8][9][19][21][22][28][29][36][37]. This success follows the general tenet of deep learning that DNNs are capable of extracting powerful features or representations from raw data in a hierarchical, layer-wise manner. However, among the many possible forms of representations in the DNN, whether some forms are more effective than others and whether their combinations are more useful are yet to be explored. Further, the highly successful DNN-HMM architecture, as carefully analyzed in [19], can be viewed as a DNN feature extractor followed by a downstream "shallow" *sequential* classifier, where each *local* classifier is a maximum-entropy one, also called softmax or multi-class logistic regression, making use of the cross-entropy training criterion. Thus, it is natural to study the relative effectiveness of a wide range of sequence classifiers beyond maximum entropy, all using the common yet diverse set of "deep" features that can be extracted from various parts of a DNN.

Earlier work has attempted to compare the use of different ways of deriving neural-network features in the Gaussian-Mixture-Model (GMM)-HMM system. The "tandem" approach makes use of the log-posterior probabilities generated by the softmax output layer of a shallow neural network [18]. The method of building deep stacking networks is a more general, recursive way of creating tandem-like features as the overall network built gets deeper and deeper [10][11][20][33]. Separately, the "bottleneck-feature" approach imposes a "bottleneck" constraint (i.e., a hidden layer with a very small number of units) in the middle of a neural network and extracts the features as a weighted sum of the outputs of the hidden layer adjacent to the bottleneck layer [16]. Deep auto-encoders are also recently used to extract bottleneck features in speech feature coding and recognition [12][27]. Most recently, Tuske et al. [31] and Yan et al. [35] experimentally compare the GMM-HMM and DNN-HMM approaches, using the tandem or top-hidden-layer features. It is reported that with the use of these DNN-derived features, the GMM-HMM recognizer can produce similar speech recognition accuracy to the DNN-HMM. If we view these recognizers as two types of "shallow" sequence classifiers both receiving DNN-derived features, then it is straightforward to understand such results, given the well understood equivalence between shallow generative models of Gaussian mixture and shallow discriminative log-linear models [17].

The work presented in this paper is focused first on a unifying view of deep learning methods for sequence classification, expressed in terms of DNN-based feature extraction followed by a (shallow) sequence classifier. Then, experimental comparisons are made among four sequence classifiers with respect to more diverse sets of DNN features than considered in the past as reviewed above. Among the four sequence classifiers, two are novel ones based on a new approach to formulating and learning recurrent neural networks (RNNs) described in detail in [3]. Among the diverse set of DNN-derived features, we include those extracted not only from the DNN output layer (e.g., as in the stacking or tandem-like approaches) but also from *all* hidden layers of the DNN as well as their combinations.

The rest of this paper is organized as follows. In Section 2, we describe diverse sets of DNN features we select to use for four types of sequence classifiers with an emphasis on the RNN classifiers. These sequence classifiers are detailed in Section 3. The experimental results are reported in Section 4, and a summary and discussion provided in Section 5.

2. THE DNN AS A FEATURE EXTRACTOR

We use Fig. 1 to illustrate how acoustic feature extraction from raw filterbank data of speech ($\boldsymbol{x}_t$) is accomplished in a DNN with three hidden layers (left), and how diverse sets of DNN-derived features are fed to the subsequent sequential classifiers (right). Here, the vector- and continuous-valued activities, $\boldsymbol{h}_{1,t}$, $\boldsymbol{h}_{2,t}$, and $\boldsymbol{h}_{3,t}$, corresponding to the three hidden layers at time frame t,

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

together with the softmax output $\boldsymbol{y}_t$ of the DNN, can be separately extracted as feature vectors for the sequence classifier. Appropriate concatenations of these DNN feature vectors, possibly with the raw speech data $\boldsymbol{x}_t$ can be used as the features to the sequence classifier as well.

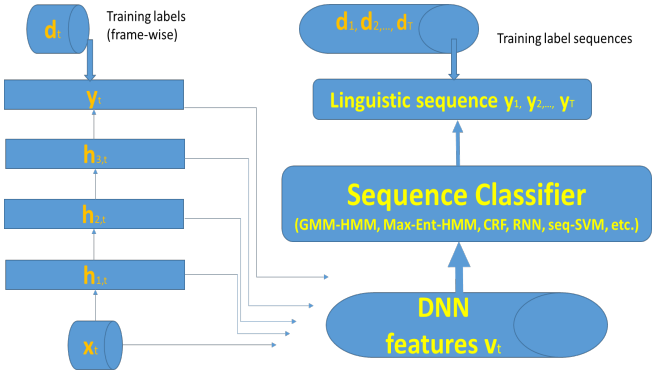

Fig. 1: *Use of the DNN as a feature extractor (left) for sequence classification (right) that produces linguistic sequences as the speech recognition system's output.*

The DNN shown on the left-hand side of Fig. 1 is trained using the one-hot coded target label vectors, denoted by $\boldsymbol{d}_t$, as the supervision signals on the frame-by-frame basis. The target vectors are constructed using one-hot coding. Temporal segmentation is required to determine $\boldsymbol{d}_t$ for each time frame t, which is accomplished in this work by forced alignment with a separate, high-quality GMM-HMM system.

Each of the DNN-derived feature vectors, $\boldsymbol{y}_t$, $\boldsymbol{h}_{1,t}$, $\boldsymbol{h}_{2,t}$, and $\boldsymbol{h}_{3,t}$, as well as their combination or concatenation can be fed as the input feature vector, denoted by $\boldsymbol{v}_t$ on the right-hand side of Fig. 1, into any type of sequence classifier including the GMM-HMM, Conditional Random Field (CRF), RNN, sequence Kernels, etc. The output of the sequence classifier is a sequence of linguistic symbols such as phrases, words, syllables, characters, or phones.

The method of learning the weight matrices in the DNN is the standard error backpropagation with the weight matrices appropriately initialized as in [5][19][23]. The objective function can be either frame-level cross-entropy or sequence-level maximum mutual information (MMI). As discussed in [24][30], these two different objective functions used for training the DNN correspond to two different types of sequence classifiers, a max-entropy classifier (followed by an HMM) and a CRF classifier, both to be discussed in the next section.

3. FOUR TYPES OF SEQUENCE CLASSIFIERS

A wide range of sequence classifiers can be used to process the DNN-derived acoustic features and to produce linguistic symbol sequences as the output of an overall speech recognition system shown in Fig. 1. To limit the scope of this work, we have experimented with four types of sequence classifiers with comprehensive phone recognition results reported in Section 4. Here we describe and discuss each of the four sequence classifiers.

3.1. Max-entropy classifier followed by an HMM

In Fig. 2 is the architecture of the Context-Dependent DNN-HMM [4][5], which uses softmax nonlinearity at the output layer of the static DNN and is trained with the frame-wise cross-entropy criterion. The DNN output gives posterior probabilities for each of many context-dependent speech classes, divided by prior class probabilities to produce (un-normalized) likelihood of the speech data for each of the HMM states [2][25].

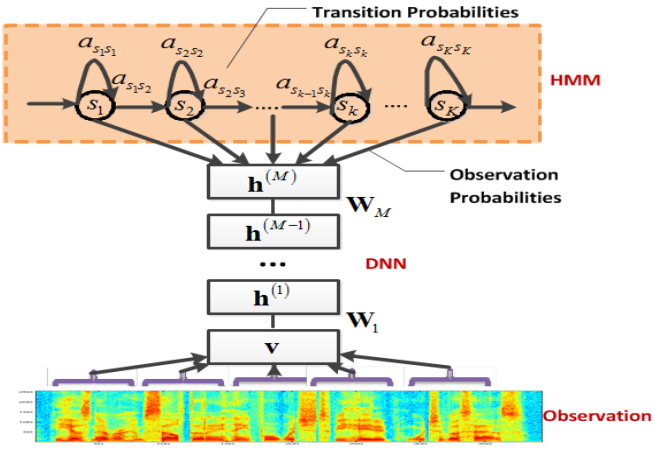

Fig. 2: *The architecture of the earliest DNN-based large-vocabulary speech recognizer* [4][5]*, which can be viewed as a DNN to extract features via locally discriminative max-entropy training), followed by an HMM as the sequential classifier.*

Let's regard the architecture of Fig. 2 as a special case of the more general family of architectures illustrated in Fig. 1, where the sequence classifier is a (static) max-entropy one followed by a (temporal/dynamic) HMM. The state posterior probabilities for the HMM correspond to the DNN output activity vector $\boldsymbol{y}_t$ in Fig. 1. Training the DNN using cross entropy is equivalent to learning a max-entropy model, a type of log-linear model. In this case, only the DNN features of $\boldsymbol{y}_t$ in Fig. 1 are used.

Another sequence classifier, closely related to the above max-entropy model, is the GMM-HMM; see [17] for the analysis of their relations. Yan et al. [35] recently reported very similar speech recognition accuracy between the use of the architecture shown in Fig. 2 and the use of the GMM-HMM as the sequence classifier. This is gratifying because of the theoretical equivalence of log-linear models (including the max-entropy model) and GMM-HMMs. As a result, we do not include in this paper the sequence classifier based on the GMM-HMM.

3.2. Conditional random field

If we train the parameters of the architecture in Fig. 2 using full-sequence MMI instead of cross entropy as the objective function, then we have an equivalence of the CRF sequence classifier that uses the output vector $\boldsymbol{y}_t$ in the DNN as the classifier's input features. This style of full-sequence training for DNN-based speech recognizers was first proposed in [24], where the equivalence was established between using MMI to perform end-to-end back-propagation training of the DNN-HMM and back-propagating errors through a liner-chain CRF to the DNN. The subsequent larger-scale full-sequence training of the DNN-HMM proved to be highly effective in reducing errors of the DNN-HMM trained with cross-entropy [22][30][32].

The CRF is used as one of four sequence classifiers in our experiments to be reported in Section 5, which confirm its superiority to the frame-based max-entropy classifier.

3.3. Recurrent neural network: the AR version

The remaining two types of sequence classifiers used in this work belong to the family of recurrent neural networks (RNNs). The autoregressive (AR) version of the RNN discussed in this section contains only the prediction from the past. Specifically, the history of the input is maintained only at the hidden states, and only the sample at the current time is input into the model, shown in Fig. 3.

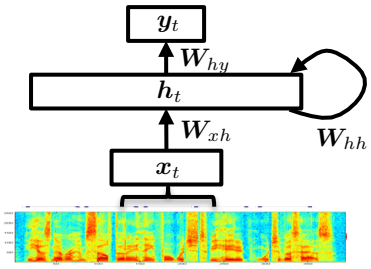

Fig. 3: *The architecture of the AR version of the recurrent neural network. The history of the input data is maintained at the hidden states and only the sample at the current time is input into the model.*

In this AR version of the RNN, the state dynamic (noise free) for the hidden state $\boldsymbol{h}_t$ is expressed mathematically as

$$\boldsymbol{h}_t = f(\boldsymbol{W}_{hh}\boldsymbol{h}_{t-1}+\boldsymbol{W}_{xh}\boldsymbol{x}_t+\boldsymbol{b}),$$

where $\boldsymbol{W}_{xh}$ and $\boldsymbol{W}_{hh}$ denote the matrices that collect the input weights and the recurrent weights, respectively, f is a nonlinear function of the hidden units (e.g., sigmoidal as is implemented in this work), and $\boldsymbol{b}$ is the vector of bias. The "observation" is the predicted "labels" or target vector, $\boldsymbol{l}_t$, a vector of one-hot coded class labels. The "observation equation" in the state space formulation becomes:

$$\boldsymbol{y}_t = \boldsymbol{W}_{hy}\boldsymbol{h}_t \quad \text{or} \quad \boldsymbol{y}_t = g(\boldsymbol{W}_{hy}\boldsymbol{h}_t)$$

where $\boldsymbol{W}_{hy}$ is the matrix of output weights, and g denotes the nonlinearity of output units (e.g., soft-max as is implemented in this work). Define the error function in model learning as a sum of squared differences between $\boldsymbol{y}_t$ and $\boldsymbol{l}_t$ over time, or cross entropy between them. Then, the method of back-propagation through time unfolds the RNN over time in computing the gradients with respect to $\boldsymbol{W}_{hy}, \boldsymbol{W}_{xh}$, $\boldsymbol{W}_{hh}$, and $\boldsymbol{b}$, and the method of stochastic gradient descent is applied to update these weight matrices and the bias vector.

3.4. Recurrent neural network: the ARMA version

Different from the AR version, the dynamics of the hidden states for autoregressive moving average (ARMA) version of the RNN is characterized by the following recursion:

$$\boldsymbol{h}_t = f\left(\boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \sum_{\tau=-\Delta_1}^{\Delta_2} \boldsymbol{W}_{xh,\tau}\boldsymbol{x}_{t-\tau} + \boldsymbol{b}\right)$$

where Δ_1 and Δ_2 denote the number of input samples that the network looks forward and backwards. If $\Delta_1 = 0$, then it only looks backwards into the past history and if $\Delta_2 = 0$, it only looks into future. When $\Delta_1 = \Delta_2 = 0$, it becomes the AR version. The first term in the parenthesis above is the AR part, and the second term in the parenthesis is the MA part. And we define the order of the moving average (MA) to be $\Delta_1 + \Delta_2 + 1$. The architecture of the ARMA version of RNN with order three is shown in Fig. 3.

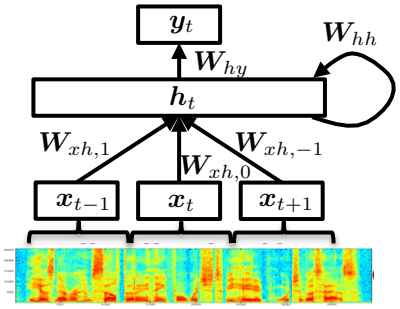

Fig. 4: *The architecture of the ARMA version of the recurrent neural network. The history of the input data is maintained both at the hidden states and the inputs' temporal context window.*

The ARMA version of RNN can be converted back into the form of AR version by defining the following extra augmented variables:

$$\overline{\boldsymbol{x}}_t \triangleq \left[\boldsymbol{x}_{t-\Delta_2}^T \quad \cdots \quad \boldsymbol{x}_{t+\Delta_1}^T\right]^T$$
$$\overline{\boldsymbol{W}}_{xh} \triangleq \left[\boldsymbol{W}_{xh,-\Delta_2} \quad \cdots \quad \boldsymbol{W}_{xh,\Delta_1}\right] .$$

Then, the dynamics for the hidden states of ARMA version of RNN become equivalent to

$$\boldsymbol{h}_t = f(\boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \overline{\boldsymbol{W}}_{xh}\overline{\boldsymbol{x}}_t + \boldsymbol{b}).$$

In other words, the ARMA version of the RNN model can be implemented in an equivalent manner by having a context window that slides through several input samples and deliver them as an augmented input sample. Thus, the same learning algorithm is used to train both versions. In this work, we apply the method of backpropagation-through-time but improve the earlier method [1][26] by formulating the training process as a formal optimization problem with an inequality constraint imposed to guarantee the stability of the RNN dynamics (see [3]).

4. EXPERIMENTS AND RESULTS

We use the TIMIT phone recognition task to evaluate the effectiveness of the diverse sets of DNN features discussed in Section 2 and the four types of sequence classifiers discussed in Section 3. The standard 462-speaker training set is used and all SA sentences are removed conforming to the standard protocol, as for example used in [7][23][24]. A separate dev set of 50 speakers is used for tuning all hyper parameters. Results are reported using the 24-speaker core test set, which has no overlap with the dev set. Signal processing for raw speech waveforms is the standard short-time Fourier transform with a 25-ms Hamming window and with a fixed 10-ms frame rate. Raw speech feature vectors are generated subsequently using filterbank analysis. This produces 41 coefficients distributed on a Mel scale (including the energy value), along with their first and second temporal derivatives.

In our experiments, 183 target class labels are used with three states for each of 61 phones. After decoding, the original 61 context-independent phone classes are mapped to a set of 39 classes for final scoring according to the standard evaluation protocol. In our experiments, a bi-gram language model over

phones, estimated from the training set, is used in decoding. The language model weight is set to one, and insertion penalty is set to zero in all experiments with no tuning.

To prepare the DNN and RNN targets during training, a high-quality tri-phone HMM model is trained on the training data set, which is then used to generate state-level labels based on HMM forced alignment.

In Table 1, we show the results of phone recognition accuracy on the TIMIT core test set for four types of sequence classifiers, all using the same labels representing 61 monophones with 3 states per phone in training. The output vectors, with a dimensionality of 183, from a DNN with three layers of 2000 hidden units each are used as the DNN features for all the four classifiers. (When 1000 hidden units are used in the DNN, the accuracy is observed to degrade slightly.) Both AR and ARMA versions of the RNN in Table 1 has a 500x500 recurrent matrix with 10% random, non-zero entries. Moving average order of the RNN (ARMA) is fixed at 13. It is clear from the results of Table 1 that the ARMA version of the RNN is the best classifier, trailed by the AR-RNN, CRF, and finally the max-entropy one.

Table 1: *Comparisons of phone recognition accuracy among four sequence classifiers, fixing the DNN-output-layer feature* $\boldsymbol{y}_t$.

Features	Max-Ent HMM	CRF	RNN(AR)	RNN(ARMA)
DNN-Out	77.9%	78.3%	80.0%	80.8%

We now fix the classifier to be an RNN (AR and ARMA versions), and assess seven types of features including four sets of DNN features (one from each DNN layer), and two ways of concatenating the above. Phone recognition accuracy results are listed in Table 2, with the symbols for the feature types being consistent with those shown in Fig. 1.

Table 2: *Phone recognition accuracy of six types of DNN-derived features, fixing the sequence classifiers to be the RNN with 200 recurrent hidden units.*

Features	RNN (AR)	RNN (ARMA)
DNN Output ($\boldsymbol{y}_t$)	79.8%	80.7%
DNN Top-Hidden ($\boldsymbol{h}_{3,t}$)	80.1%	81.0%
DNN Top-Hidden ($\boldsymbol{h}_{2,t}$)	79.5%	79.9%
DNN Top-Hidden ($\boldsymbol{h}_{1,t}$)	77.0%	78.3%
Raw Filterbank Data ($\boldsymbol{x}_t$)	69.6%	71.8%
Concatenating [$\boldsymbol{h}_{3,t}$ $\boldsymbol{y}_t$]	80.2%	81.2%
Concatenating [$\boldsymbol{h}_{3,t}$ $\boldsymbol{x}_t$]	79.7%	80.4%

The most striking observation from Table 2 is that the high-level feature extracted from the top DNN hidden layer, $\boldsymbol{h}_{3,t}$, is the most effective, followed by the "tandem" feature $\boldsymbol{y}_t$ derived from the DNN output layer. Lower DNN hidden layers provide less effective features but they are still much better than the raw filterbank feature $\boldsymbol{x}_t$ feeding directly to the RNN. A simple account for the difficulty of the RNN receiving raw acoustic data as its input is the temporal *non-smooth* nature of the data, which is hard for the RNN to use the parameters in the recurrent matrix to predict from one frame to the next. After the DNN extracts high-level features from raw acoustic data, the RNN can more easily predict the temporally smooth high-level features. The best accuracy, 81.2%, is obtained using concatenated features from the top hidden layer and output layer of the DNN as the input to the RNN (ARMA with order 13), with the total input dimensionality of 2183. This reaches almost the same level of the highest accuracy for this task achieved by the deep convolutional net [7] or by the deep-RNN (with LSTM) [14][15], and we do so without the delicate design of the convolution-pooling structure as required in [7] and without any special memory structure as required in [14][15].

The high accuracy performance achieved by the use of DNN features and the subsequent RNN (ARMA) classifier is very robust. In Fig. 5 is a typical curve on the progressively improved test-set phone recognition accuracy as the RNN training epochs advance.

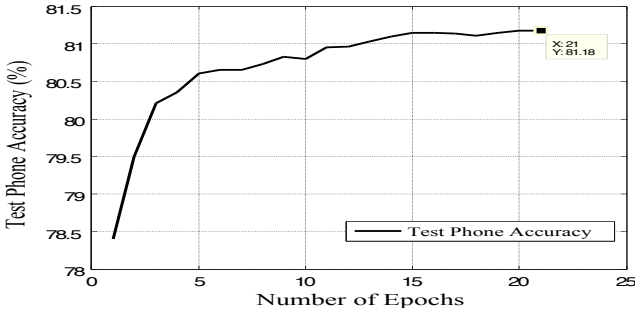

Fig. 5: P*hone recognition accuracy on the TIMIT core test set as a function of the epochs during the RNN (ARMA) training.*

5. DISCUSSIONS AND FUTURE WORK

In this paper, we develop a unifying scheme for deep learning the speech-centric sequence classification problems. As illustrated in Fig. 1, this scheme breaks the problem solution into two parts: 1) The use of (static) deep architectures (e.g., the DNN) to extract high-level acoustic features that are temporally much more smooth than the raw acoustic data sequence; and 2) The use of a sequence classifier performing the dynamic decoding task to derive linguistic sequences. Compared with end-to-end learning, this two-stage approach appears to have the benefit of regularization. It helps overcome the problem of overfitting in sequence classification tasks using the deep models with a very high capacity. Such an overfitting problem was found to be prominent in the studies reported in [24][30].

While the highly positive evaluation of this scheme has so far been carried out on a phone recognition task, where phone sequences are the output symbols, we expect it to be effective for other tasks such as continuous word recognition, machine translation, and spoken language understanding, where either word sequences or semantic-slot sequences would become the output of the overall deep learning system.

Future work will aim to improve the already powerful sequence classifier of RNN (ARMA). Effective "pre-training" placed within the frameworks of advanced optimization [34] and joint generative-discriminative modeling [6][13] is expected to enhance the performance of RNN-based sequence classification using distributed hidden-state representations and nonlinear state-space formulation of speech dynamics on the DNN features.

12. REFERENCES

[1] Bengio, Y., Boulanger-Lewandowski, N., and Pascanu, R. "Advances in optimizing recurrent networks," Proc. ICASSP, 2013.

[2] Bourlard H. and Morgan, N.. Connectionist Speech Recognition: A Hybrid Approach, Kluwer, 1993.

[3] Chen, J. and Deng, L. "A Primal-Dual Method for Training Recurrent Neural Networks Constrained by the Echo-State Property", Proc. ICLR, 2014.

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

[5] Dahl, G., Yu, D., Deng, L., and Acero, A. "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," IEEE Trans. on Audio, Speech, and Language Processing, vol. 20, pp. 30–42, 2012.

[6] Deng, L. and Li, X. "Machine learning paradigms in speech recognition: An overview," IEEE Transactions on Audio, Speech, & Language, vol. 21, pp. 1060 – 1089, May 2013.

[7] Deng, L., Abdel-Hamid, O., and Yu, D. "A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion," Proc. ICASSP, 2013.

[8] Deng, L., Hinton, G., and Kingsbury, B. "New types of deep neural network learning for speech recognition and related applications: An overview," Proc. ICASSP, 2013a.

[9] Deng, L., Li, J., Huang, K., Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero. "Recent advances in deep learning for speech research at Microsoft," Proc. ICASSP, 2013b.

[10] Deng, L., Yu, D., and Platt, J. "Scalable stacking and learning for building deep architectures," Proc. ICASSP 2012.

[11] Deng, L. and Yu, D. "Deep Convex Network: A scalable architecture for speech pattern classification," Proc. Interspeech, 2011.

[12] Deng, L., Seltzer, M., Yu, D., Acero, A., Mohamed, A., and Hinton, G. "Binary coding of speech spectrograms using a deep auto-encoder," Proc. Interspeech, 2010.

[13] Deng, L. and O'Shaughnessy, D. SPEECH PROCESSING – A Dynamic and Optimization-Oriented Approach, Marcel Dekker, 2003.

[14] Graves, A., Mohamed, A., and Hinton, G. "Speech recognition with deep recurrent neural networks," Proc. ICASSP, 2013.

[15] Graves, A., Jaitly, N., and Mohamed, A. "Hybrid speech recognition with deep bidirectional LSTM," Proc. ASRU, 2013a.

[16] Grezl, F. Karafiat, M., Kontar, S., and Cernocky, J. "Probabilistic and bottleneck features for LVCSR of meetings," Proc. ICASSP, 2007.

[17] Heigold, G., Ney, H., Lehnen, P., Gass, T., Schluter, R. "Equivalence of generative and log-liner models," IEEE Trans. Audio, Speech, and Language Proc., vol. 19, February 2011, pp. 1138-1148.

[18] Hermansky, H, Ellis, D. and Sharma, S. "Tandem connectionist feature extraction for conventional HMM systems," Proc. ICASSP, 2000.

[19] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. "Deep neural networks for acoustic modeling in speech recognition" IEEE Signal Processing Magazine, vol. 29, November 2012, pp. 82–97.

[20] Hutchinson, B., Deng, L., and Yu, D. "Tensor deep stacking networks," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 35, 2013, pp. 1944-1957.

[21] Jaitly, N., Nguyen, P., Senior, A. and V. Vanhoucke, "Application of pre-trained deep neural networks to large vocabulary speech recognition," in Proc. InterSpeech-2012,

[22] Kingsbury, B., Sainath, T., and Soltau, H. "Scalable minimum Bayes risk training of deep neural network acoustic models using distributed Hessian-free optimization," Proc. Interspeech, 2012.

[23] Mohamed, A., Dahl, G. and Hinton, G. "Acoustic modeling using deep belief networks", IEEE Transactions Audio, Speech, & Language Proc. Vol. 20. No. 1, January 2012.

[24] Mohamed A., Yu, D., and Deng, L. "Investigation of full-sequence training of deep belief networks for speech recognition," Proc. Interspeech, 2010.

[25] Morgan, N. "Deep and wide: Multiple layers in automatic speech recognition," IEEE Transactions on Audio, Speech, and Language Proc., vol. 20, January 2012.

[26] Pascanu, R., Mikolov, T., and Bengio, Y. "On the difficulty of training recurrent neural networks," Proc. ICML, 2013.

[27] Sainath, B. Kingsbury, and B. Ramabhadran, "Auto-encoder bottleneck features using deep belief networks," Proc. ICASSP, 2012,

[28] Sainath, T., Kingsbury, B., Soltau, H., and Ramabhadran, B., "Optimization techniques to improve training speed of deep neural networks for large speech tasks, IEEE Transactions on Audio, Speech, and Language Processing, vol.21, Nov. 2013, pp. 2267-2276.

[29] Seide, F., Li, G., and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," Proc. Interspeech, 2011.

[30] Su, H., Li, G., Yu, D., and Seide, F. "Error back propagation for sequence training of context-dependent deep neural networks for conversational speech transcription," Proc. ICASSP, 2013.

[31] Tuske, Z. Sundermeyer, M. Schluter, R. and H. Ney, "Context dependent MLPs for LVCSR: Tandem, hybrid or both?" Proc. Interspeech, 2012.

[32] Vesely, K, Ghoshal, A., Burget, L., Povey, D. "Sequence-discriminative training of deep neural networks," Proc. ICASSP, 2013.

[33] Vinyals, O., Jia, Y., Deng, L., and Darrell, T. "Learning with recursive perceptual representations," Proc. Neural Information Processing Systems (NIPS), vol. 15, 2012.

[34] Wright, S., Kanevsky, D., Deng, L., He, X., Heigold, G., Li, H. "Optimization Algorithms and Applications for Speech and Language Processing," IEEE Transactions on Audio, Speech, and Language Processing, vol.21, no.11, pp.2231-2243, 2013.

[35] Yan, Z., Huo, Q, and Xu, J. "A scalable approach to using DNN-derived features in GMM-HMM based acoustic modeling for LVCSR," Proc. Interspeech, 2013.

[36] Yu, D., Deng, L., and Seide, F. "The deep tensor neural network with applications to large vocabulary speech recognition," IEEE Transactions on Audio, Speech, and Language Processing, vol. 21, no. 2, pp. 388-396, 2013.

[37] Yu, D., Deng, L., and Dahl, G. "Roles of Pre-Training and Fine-Tuning in Context-Dependent DBN-HMMs for Real-World Speech Recognition," Proc. NIPS Workshop, 2010.