DEEP LEARNING OF SPLIT TEMPORAL CONTEXT FOR AUTOMATIC SPEECH RECOGNITION

Moez Baccouche, Benoît Besset, Patrice Collen and Olivier Le Blouch

Orange Labs - France Télécom, 4 rue du Clos Courtel,
F-35510, Cesson-Sévigné, France.

ABSTRACT

This paper follows the recent advances in speech recognition which recommend replacing the standard hybrid GMM/HMM approach by deep neural architectures. These models were shown to drastically improve recognition performances, due to their ability to capture the underlying structure of data. However, they remain particularly complex since the entire temporal context of a given phoneme is learned with a single model, which must therefore have a very large number of trainable weights. This work proposes an alternative solution that splits the temporal context into blocks, each learned with a separate deep model. We demonstrate that this approach significantly reduces the number of parameters compared to the classical deep learning procedure, and obtains better results on the TIMIT dataset, among the best of state-of-the-art (with a 20.20% PER). We also show that our approach is able to assimilate data of different nature, ranging from wide to narrow bandwidth signals.

Index Terms— Speech recognition, neural networks, deep learning, split temporal context.

1. INTRODUCTION

The use of artificial neural networks (ANNs) for automatic speech recognition began several decades ago, mainly in hybrid systems combining ANNs with Hidden Markov Models (HMMs) [1]. The key idea of such approaches is to train a neural model to estimate posterior states of the HMM, and thus to compute the acoustic emission probabilities using Bayes rule. However, despite their early promise, ANN/HMM systems were surpassed by other hybrid models combining HMM with Gaussian Mixture Models (GMMs) [2], which were shown to be more accurate, and can be trained with several discriminative and "easy to implement" techniques.

However, despite these advantages, GMM/HMM approaches suffer from many drawbacks, especially when dealing with non-linear modeling problems. Thus, it has long been suspected that neural models could outperform GMM/HMM approaches, if the standard one-layer neural architecture was replaced by more complex ones, with many more trainable parameters. Yet, such complex models are difficult to train, and have serious over-fitting shortcomings. Recent advances in machine learning offered some solutions to these problems. The first one relies on so-called deep models, i.e. carefully designed and learned neural architectures with several hidden layers. With the recent methods based on layer-by-layer greedy training, such deep architectures can be effectively learned, which perform extremely well not only on the TIMIT dataset [3] but also on large vocabulary tasks [4, 5, 6, 7] leading to wide industrial adoption of deep learning in speech recognition.

Other recent works proposed different solutions, which can be summarized as reporting the complexity on the design of the neural system structure rather than increasing the number of trainable parameters. Indeed, several works found that using hierarchical architectures, which learn each part of the signal with a different model, can be beneficial for speech recognition. For example, the TRAP system [8] proposes to learn temporal segments of feature vectors corresponding to critical bands spectral densities. A separate neural classifier is trained with data coming from each critical band, and obtained outputs are then used to feed another neural classifier which function is to combine all decisions into a final one. Another similar approach, called Split Temporal Context (STC) was introduced by Schwarz et al. [9], and proposes a different hierarchical structure operating on a shorter temporal context window, and yielding better results.

Nevertheless, both TRAP and STC systems use simple neural classifiers with a single hidden layer. In this paper, we propose to take benefits of the deep architectures high modeling power, by introducing an approach which combines the structural characteristics of the STC system with the recent deep learning techniques.

The rest of the paper is organized as follows. Section 2 describes an overview of the proposed model, and the corresponding training procedure. We present in section 3 experimental results on the TIMIT [3] and NTIMIT [10] datasets, before concluding and giving some perspectives of this work in section 4. Finally, section 5 discusses how the contributions presented in this paper are related to prior work in the field.

2. PROPOSED APPROACH

2.1. The model

The proposed model is illustrated on Figure 1. It can be seen as a combination of the STC system of Schwarz et al. [9] with deep learning techniques [6]. The idea is to take benefits of the ability of deep models to capture the underlying structure of data, while simplifying the modeling task by operating on temporally short context windows instead of the long non-split ones.

Our approach consists in three steps: (i) split the long temporal context into blocks as in the STC system, (ii) model each block with a separate deep neural network, and (iii) a final step in which a neural network is trained to *merge* individual decisions corresponding to each block.

Concretely, the system operates on a set of B critical-bands corresponding to a Fourier transform filter-bank and a long temporal context of L frames (see Figure 1). The speech contained in the current context window is thus encoded in a set of L feature vectors (one per frame) having B coefficients each, and describing a segment of temporal evolution of the critical-bands spectral densities.

As mentioned above, the temporal context window is split into clusters, called STC blocks, with one overlapping frame between

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

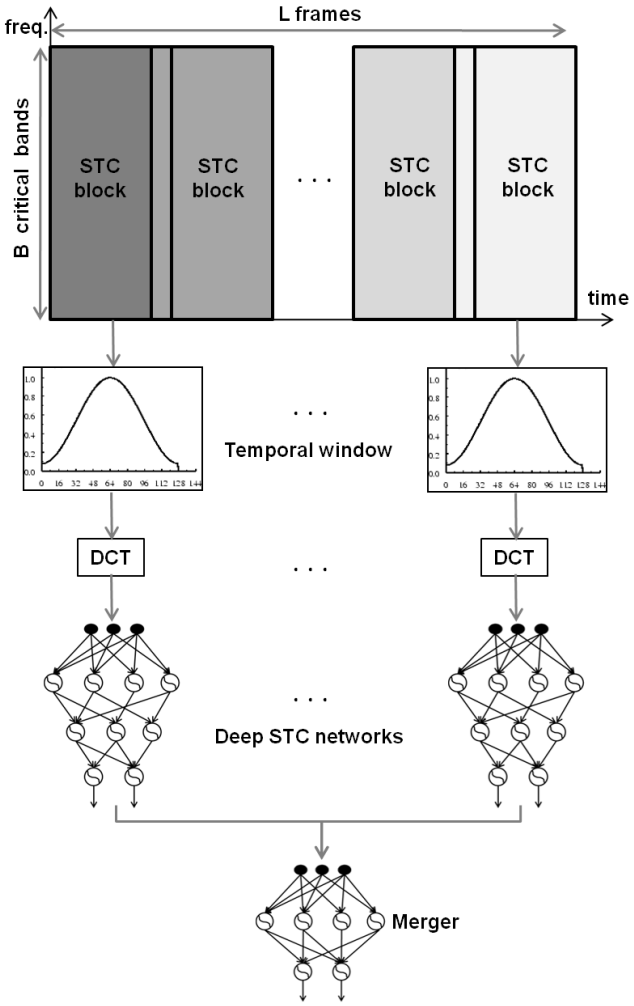

Fig. 1. Overview of the proposed approach.

each two consecutive blocks. Each STC block undergoes a temporal weighting step followed by a *Discrete Cosine Transform* (DCT). As in [6], obtained vectors are normalized in order to have zero mean and unit variance. These normalized feature vectors constitute the representation level on which learning is performed: for each STC block, a deep neural model (called *STC network*) is trained to output a probability distribution over the possible phoneme labels. These labels correspond to the commonly used HMM-based states representation in speech modeling, in which each phoneme is associated with 3 hidden states. Note that for each training sample, all *STC networks* are trained to target the same label: the one corresponding to the central frame of the complete (non-split) temporal context window.

Obtained outputs are collected from each STC network and concatenated to generate a vector which encodes individual decisions coming from each block. This vector is normalized, as described previously for STC networks, and used to train another neural network (called *merger*), which learns to estimate the final posterior distribution over HMM states, by fusing all decisions. The *merger* outputs are thus fed into the Viterbi decoder, as for a classical HMM-based speech recognition system.

2.2. The training procedure

All deep neural models used in our system are trained with the procedure described in [6], in which a first generative *pre-training* step is followed by a discriminative *fine-tuning* one. The idea is to replace the standard *random* initialization of the network trainable weights by a better *starting point*, which is learned directly from the input data. This initialization has been shown to lead to faster convergence and to prevent from over-fitting [11].

Note that, in addition to phoneme recognition, this type of training procedure, which combines unsupervised and supervised learning, has been widely and successfully used in a variety of other machine learning problems, ranging from character recognition [12] to information retrieval [13].

The *pre-training* step is layer-wise, i.e. it consists in learning one pair of layers at a time, with the internal states of this pair acting as the data for training the next one. Each learning module (i.e. a pair of layers) is called a *Restricted Boltzmann Machine* (RBM): an undirected graph formed by visible and hidden units, that models respectively observations and features. Two kinds of RBMs are used: (i) Gaussian RBMs, i.e. which allows real-valued states for its visible units, and (ii) binary RBMs, i.e. with only binary units. Concretely, an input Gaussian RBM is first trained with the real-valued feature vectors. Then, obtained binary hidden units are used as data for training the next RBM. This is repeated to learn as many pairs of layers as needed. Note that Gaussian RBMs are used only for the input layer. Training is performed using the contrastive divergence algorithm [14]. Once all layers are trained, they are stacked, and an output layer is added to form the final multi-layer (deep) architecture.

Fine-tuning consists in performing a standard back-propagation with *momentum* algorithm, initializing the network with the values learned during the *pre-training* step. This allows to slightly adjust the weights in order to approach the most discriminative weight-space region. We used the classification error rate per frame as objective function during this training step. Note that the hole *fine-tuning* phase is repeated several times (5 times in our experiments, -see section 3-) to refine the HMM states alignment over the phoneme temporal segment (with the first iteration corresponding to an uniform split into three segments of equal lengths). Note also that only deep architectures are trained using this two-steps procedure, neural networks with only one hidden layer (for example the *merger* in our case, as we will see in section 3) are simply trained with a standard back-propagation with a *random* initialization.

3. EXPERIMENTAL RESULTS

In this section, we present the experimental results corresponding to the proposed approach described above. First, we focus on the evaluation of the recognition performance of our system when operating on wide band signals. This will be done using the TIMIT dataset [3]. Then, we will investigate the possibility of using our approach to process data in both narrow and wide bands.

3.1. Experiments on the TIMIT dataset

The TIMIT *Acoustic-Phonetic Continuous Speech Corpus* [3] is a standard dataset used for speech recognition. It consists of 630 speakers reading 10 sentences each. These sentences are phonetically rich, and represent 8 *American English* dialects. We used the standard training set (corresponding to 462 speakers, after removing the *speaker calibration* sentences) and a separate validation set

to: (i) tune the hyper-parameters, and (ii) perform the *early stopping* procedure (i.e. stopping the training algorithm before over-fitting).

Results are computed using the standard core test set, which has no overlap with the training and validation sets. The evaluation is performed on the phone level (with the standard CMU/MIT phone mapping [15]), based on the supplied phone transcription, and using the *Phone Error Rate* (PER) metric. The optimal number of substitutions, deletions and insertions during the dynamic programming alignment was tuned on the validation set.

In all our experiments, the number of critical-bands B was set to 23. Note that we have also experimented the use of a higher number of coefficients per frame (typically 40, with their first and second temporal derivatives, as recommended in [6]), but we observed that this increases considerably the complexity of the model (since *STC networks* would have larger input layers) without improving its performance.

We used a long temporal context corresponding to $L = 31$ frames: the central actual frame to be recognized, 15 frames in past and 15 in future. The input of our system (before splitting the temporal context and applying temporal windowing and DCT) is therefore a vector containing 713 values (31 frames $\times$ 23 coefficients).

Regarding the training, for both *pre-training* and *fine-tuning* steps we used the *Neural Network Trainer TNet* library, which proposes a CUDA GPGPU implementation of the mini-batch back-propagation and the contrastive divergence algorithms. The training was performed using these parameters:

- For the *pre-training* step: learning rates of 5×10^{-3} and 8×10^{-2} respectively for the Gaussian and binary RBMs, a momentum of 0.9 (except for the 5 first epochs, which use no momentum) and a mini-batch size of 128.
- For the *fine-tuning* step: an initial learning rate of 8×10^{-3} (which is halved every epoch for which the validation PER decreases by less than 0.5), a momentum of 0.5 and a mini-batch size of 512.

Note that the *pre-training* phase was performed using only a small subset of the training dataset, which was shown to considerably reduce the computation time without affecting performance. Note also that an early stopping procedure was used during *fine-tuning* which consists in interrupting the training when the validation PER decreases by less than 0.1 after halving the learning rate.

We have performed a set of experiments varying the different architectural parameters of our recognition system in order to select the optimal ones. We have experimented: (i) three types of temporal windows (Hamming, rectangular and the 0.24×1.1^x windows), (ii) applying or not a DCT which keeps 5 coefficients per STC block, (iii) varying the number of STC blocks (2, 3 or 5) and (iv) several *STC networks* architectures. Concerning the latter, deep models (with more than one layer) were pre-trained as described in section 2, while architectures with only one layer were directly trained with back-propagation algorithm. Regarding the *merger* network, it corresponds to a one layer architecture with 1500 neurons for all the experiments. Other *merger* architectures (including deep ones) were also tested with no performance improvement.

We report on Table 1 obtained results, corresponding to the PER on the test core set obtained in the last (fifth) iteration of the *fine-tuning* step. The best model, yielding a PER of 20.20, corresponds to a split of the temporal context into 5 blocks, with a three-layer deep architecture for each STC block. The best results are obtained with a 5 coefficients DCT, and without temporal windowing, which is consistent with the observations made by Schwarz et al. in [9].

Temporal window	DCT size	#STC blocks	STC Networks architecture	PER (%)
Hamming	5	2	1500×1	22.73
Hamming	5	3	1500×1	22.40
Hamming	5	5	1500×1	21.81
0.24×1.1^x	5	5	1500×1	21.82
Rectangular	5	5	1500×1	21.70
Rectangular	5	5	200×3	20.71
Rectangular	5	5	1000×3	20.38
Rectangular	5	5	500×3	**20.20**
Rectangular	–	5	500×3	20.88

Table 1. Evaluation of the obtained PER on the TIMIT test set for different architectural parameters of the proposed model. PER values correspond to the last (fifth) iteration of the *fine-tuning* step. For the *STC networks* architecture, the notation $n \times l$ emphasizes an architecture of l hidden layers with n neurons per layer.

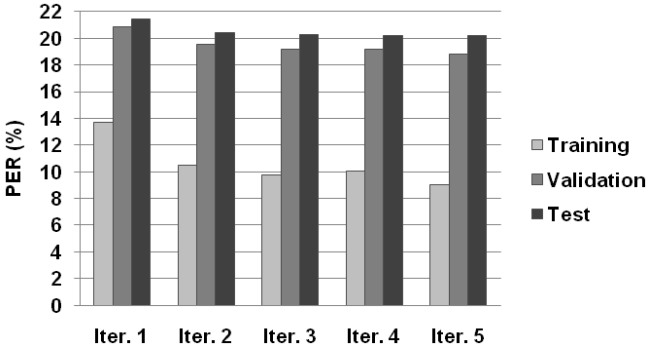

Fig. 2. Detailed results per iteration obtained by the proposed model on the TIMIT corpus. The best result (corresponding to a PER of 20.20% on the test core set) is obtained in the fifth iteration.

We depict in Figure 2 the detailed results of our best architecture (i.e. the PER values per iteration obtained on the training, validation and test sets). One can observe that the HMM states realignment procedure (performed from the second iteration) significantly improves performances compared to the first iteration for which each phoneme temporal segment is split uniformly. The best result, corresponding to the best PER on the validation set, is obtained for the last (fifth iteration).

The architecture obtaining the best results contains approximately 4.18×10^6 trainable weights (about 620.000 for each *STC network*, and 1.08×10^6 for the *merger*). Table 2 shows the computation time (more precisely the number of epochs required for convergence, and the computation time of each epoch) needed to perform the last iteration of the *fine-tuning* step for this architecture using a machine with a quad-core 3.6 GHz CPU and a NVIDIA Quadro 600 GPU.

Regarding the *pre-training* step, it took about 11 minutes for each STC block using the same machine. Thus, the complete training scheme (i.e. *pre-training* of each *STC network*, followed by 5 iterations of *fine-tuning*) needs approximately 10 hours of computation. The testing phase takes a lot less time since all the TIMIT corpus (including training, validation and test sets) is decoded in about 8 minutes.

In order to evaluate the relevance of our results, we show on Table 3 a comparison with the state-of-the-art on the TIMIT dataset.

Network	#epochs	Time / epoch
STC network 1	18	
STC network 2	18	
STC network 3	19	58 sec
STC network 4	17	
STC network 5	13	
Merger	18	1 min 15 sec

Table 2. Computation time corresponding to the last (fifth) iteration of the *fine-tuning* training step.

We also report the number of trainable parameters corresponding to each approach to evaluate their complexity. Table 3 shows that our model obtains performances among the best of related work. More particularly, we obtain better results than the STC [9] and the deep learning [6] approaches, showing that the combination of both of them improves recognition performances. Furthermore, our model is approximately 9 times less complex than the one operating on a non-split temporal context, as described by Mohamed et al. in [6].

Method	#param.	PER (%)
CRF [16]	4500	34.77
Augmented CRF [17]	0.02 M	27.30
CD-HMM [18]	4.00 M	26.60
Recurrent NN [19]	0.10 M	26.10
HTM [20]	0.01 M	24.83
STC [9]	1.50 M	21.48
RBM deep model [6]	37.7 M	20.70
mcRBM deep model [5]	20.9 M	20.50
STC + RBM deep models (ours)	**4.18 M**	**20.20**
Deep ConvNets NN [21]	4.03 M	20.07
Deep ConvNets NN + pooling [22]	-	18.70
Deep LSTM recurrent NN [23]	4.30 M	17.70

Table 3. Comparison of obtained results on the TIMIT test set with related work.

Up to our knowledge, only the recent works by Abdel-Hamid et al. [21], Deng et al. [22] and Graves et al. [23], based respectively on deep ConvNets (with or without pooling) and LSTM recurrent neural networks, obtain better results than ours. For the latter, The outstanding performances can be explained by the demonstrated ability of the LSTM model to deal with very long temporal context (the entire sequence vs 31 frames used in this work). Note that almost all approaches mentioned in Table 3 use a language model as a post-processing in order to improve the performance of their systems (for example a 0.34 point improvement in [9]), which is not our case. Another interesting observation is that all approaches obtaining a PER inferior to 21 use deep neural architectures.

3.2. Experiments on the NTIMIT dataset

We evaluate in this subsection the performance of the proposed model described above on wide and narrow (telephonic) bandwidths. The aim is to be able to model both types of signals with a single recognition system, which is particularly interesting (mainly in an industrial context) and has a high applicative potential.

As in [24, 25], the model was trained/tested using two datasets: the TIMIT dataset for the wide bandwidth, and the NTIMIT (Network TIMIT) dataset [10] for the narrow bandwidth. The latter was collected by transmitting the TIMIT corpus over the telephone network. The experiment consists in training the model with the two datasets together or with each of them separately, and then test it on both datasets. Corresponding results, obtained using the same training parameters and evaluation protocol described above, are reported on Table 4.

Training dataset	Test on TIMIT	Test on NTIMIT
TIMIT + NTIMIT	20.37	29.55
TIMIT	20.20	68.20
NTIMIT	32.97	31.72

Table 4. Obtained PERs on wide (TIMIT) and narrow (NTIMIT) bandwidth test sets using mixed bandwidths training data.

Table 4 shows that training the model with both datasets leads to better results on NTIMIT (with 2.17 points of improvement), and slightly worse results on TIMIT (with 0.17 point of deterioration). This can be explained by the fact that wide bandwidth signals contain richer information, which permits to remove some confusions. On the other side, adding narrow bandwidth data during the training introduces noise, which reduces the recognition performance. Finally, the results for each type of signals are still satisfactory and show that the proposed model can assimilate data from different nature.

4. CONCLUSION AND FUTURE WORK

In this paper, we have presented a neural approach for acoustic modeling in speech recognition, which combines the hierarchical STC method with deep learning techniques. We have demonstrated that using several "*small*" deep architectures to model short temporal windows is significantly less complex and gives better results than learning the entire context with a single "*huge*" model. Experimental results, obtained on the TIMIT dataset, confirm the high performance of our approach since it achieves results among the best of related work (with a PER of 20.20% on the test set). We have also demonstrated that the hierarchical nature of our model enables it to assimilate wide and narrow bandwidths signals, which is particularly interesting for a wide range of applications.

Future work will address the application of our model to very large corpus. We are currently performing a set of experiments on a french corpus containing about 120 hours of wide and narrow bandwidths speech data collected from 1000 speakers. Preliminary results are quite satisfactory since they correspond to a PER of 12.53%.

5. RELATION TO PRIOR WORK

The work presented in this paper focused on neural-based speech recognition systems, and more precisely on those having an hierarchical structure, i.e. which model the speech by a hierarchy of networks, each one for a specific part of the signal.

The proposed approach is closely related to several prior works which have addressed this issue. For example the TRAP model, introduced by Hermansky and Sharma [8] (and its different variants [26, 27]), or the STC system by Schwarz et al. [9], which obtains better results with a shorter temporal context. Our model extends the work of Schwarz et al. [9] by incorporating deep learning techniques, taking benefits from the recent advances in this field [6] in order to boost recognition performances.

Thus, we propose to use simultaneously two types of hierarchical modeling: (i) the block-based processing of the speech signal proposed by the STC system, and (ii) the layer-wise representation of the information using deep architectures.

6. REFERENCES

[1] N. Morgan and H. Bourlard, "Continuous speech recognition using multilayer perceptrons with hidden markov models," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1990.

[2] B.-H. Juang, S. Levinson, and M. Sondhi, "Maximum likelihood estimation for multivariate mixture observations of markov chains," in *IEEE Transactions on Information Theory*, 1986.

[3] W. Fisher, G. Doddington, and K. Goodie-Marshall, "The DARPA speech recognition research database: Specifications and status," in *DARPA workshop on Speech Recognition*, 1986.

[4] D. Yu, L. Deng, and G. E. Dahl, "Roles of pre-training and fine-tuning in context-dependent dbn-hmms for real-world speech recognition," in *NIPS workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[5] G. Dahl, A.-R. Mohamed, and G. E. Hinton, "Phone recognition with the mean-covariance restricted boltzmann machine," in *Advances in neural information processing systems*, 2010.

[6] A.-R. Mohamed, G. Dahl, and G. E. Hinton, "Acoustic modeling using deep belief networks," in *IEEE Transactions on Audio, Speech, and Language Processing*, 2012.

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

[8] H. Hermansky and S. Sharma, "Temporal patterns (TRAPS) in ASR of noisy speech," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1999.

[9] P. Schwarz, P. Matejka, and J. Cernocky, "Hierarchical structures of neural networks for phoneme recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2006.

[10] C. Jankowski, A. Kalyanswamy, S. Basson, and J. Spitz, "NTIMIT: a phonetically balanced, continuous speech, telephone bandwidth speech database," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1990.

[11] 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 *International conference on Machine learning*, 2007.

[12] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, 2007.

[13] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," in *Science*, 2006.

[14] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," in *Neural computation*, 2002.

[15] K.-F. Lee and H.-W. Hon, "Speaker-independent phone recognition using hidden markov models," in *IEEE Transactions on Acoustics, Speech and Signal Processing*, 1989.

[16] J. Morris and E. Fosler-Lussier, "Combining phonetic attributes using conditional random fields.," in *Annual Conference of the International Speech Communication Association*, 2006.

[17] Y. Hifny and S. Renals, "Speech recognition using augmented conditional random fields.," in *IEEE Transactions on Audio, Speech, and Language Processing*, 2009.

[18] L. Lamel and J.-L. Gauvain, "High performance speaker-independent phone recognition using CDHMM," in *European Conference on Speech Communication and Technology*, 1993.

[19] T. Robinson, M. Hochberg, and S. Renals, "The use of recurrent neural networks in continuous speech recognition," in *Automatic speech and speaker recognition*. Springer, 1996.

[20] L. Deng and D. Yu, "Use of differential cepstra as acoustic features in hidden trajectory modeling for phonetic recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2007.

[21] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2012.

[22] L. Deng, O. Abdel-Hamid, and D. Yu, "A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

[23] A. Graves, A.-R. Mohamed, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2013.

[24] J. Li, D. Yu, J.-T. Huang, and Y. Gong, "Improving wideband speech recognition using mixed-bandwidth training data in CD-DNN-HMM," in *IEEE Spoken Language Technology Workshop*, 2012.

[25] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, and J. Williams, "Recent advances in deep learning for speech research at Microsoft," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2013.

[26] P. Jain and H. Hermansky, "Beyond a single critical-band in TRAP based ASR.," in *Annual Conference of the International Speech Communication Association*, 2003.

[27] B. Y. Chen, Q. Zhu, and N. Morgan, "Learning long-term temporal features in LVCSR using neural networks.," in *Annual Conference of the International Speech Communication Association*, 2004.