ISCA Archive
http://www.isca-speech.org/archive

INTERSPEECH 2012
ISCA's 13th Annual Conference
Portland, OR, USA
September 9-13, 2012

Integrating Deep Neural Networks into Structured Classification Approach based on Weighted Finite-State Transducers

Yotaro Kubo, Takaaki Hori, Atsushi Nakamura

NTT Communication Science Laboratories, Kyoto, Japan
{kubo.yotaro, hori.t, nakamura.atsushi}@lab.ntt.co.jp

10.21437/Interspeech.2012-14

Abstract

Recently, deep neural networks (DNNs) have been drawing the attention of speech researchers because of their capability for handling nonlinearity in speech feature vectors. On the other hand, speech recognition based on structured classification is also considered important since it realizes the direct classification of automatic speech recognition. For example, a structured classification method based on weighted finite-state transducers (WFSTs) introduces a linear classification term for each arc transition cost in a decoding network to capture contextual information from decoder states. In this paper, we focus on the integration of a WFST-based structured classifier and DNNs. Since these two approaches attempt to improve the representation of features and labels, respectively, the combination of these models would be efficient because of their complementarity. In the proposed method, DNNs are used to extract discriminative features, and then the features are classified by using WFST-based structured classifiers. The proposed method is evaluated by using TIMIT continuous phoneme recognition tasks. We confirmed that combining structured classification leads to stable performance improvements even from the well-optimized deep neural network acoustic models.

Index Terms: Speech recognition, deep neural networks, structured classification, weighted finite-state transducers

1. Introduction

Deep neural networks (DNNs) and their noteworthy performance in several speech recognition experiments have been attracting attention. These methods involve the classification of acoustic features into pre-defined HMM-states by using neural networks with many layers [1, 2]. The motivation behind this deep hierarchical processing is that the nonlinear warping required for classifying speech features is assumed to have a hierarchical structure. [3] mentions that the number of hidden units that are needed to correctly model a hierarchical process by using single hidden layer neural networks is increasing exponentially. Since such single layer networks with extremely large hidden units tend to be overfitted, the use of deep and relatively narrow hierarchical networks is assumed to be important. In that sense, we can say that one of the advantages of deep neural networks is its capacity to transform feature vectors into linearly separable vectors. Furthermore, recently it has been suggested that the hidden activation of DNNs is successfully suppressing the speaker variability [4].

On the other hand, flat and direct approaches that can directly optimize decoder behavior by integrating several lexical and acoustical features have also been considered important. For example, [5] provides a flexible framework that can integrate several detectors to realize accurate recognition. [6] introduced a joint representation of language models and acoustic models by using conditional random fields. These method maintain theoretical validity by avoiding the nonlinear warping of the features, and only treating linear classification. Although assuming linear classification is not realistic, such linear classifiers, in practice, achieve higher performances by expanding representation of target classes. For example, in [7], the structured classifier is designed to classify each arc in a decoding network represented as weighted finite-state transducers (WFSTs). In that case, since the arcs of the WFSTs represent hidden states of hidden Markov models (HMMs), phonemes, positions in words, and words, the number of target classes is exponentially increasing. The expansion of target classes is important since the classification boundaries of linear classifiers can be written as piece-wise linear functions where the number of pieces is proportional to the number of target classes. Therefore, one of the main advantages of the structural classification approach is the expanded representation of target classes.

In this paper, we attempt to expand the target classes of DNN acoustic models by exploiting the structure of WFST decoders as in [7]. The expansion of target classes is also considered important in DNN-based speech recognition systems. For example, [8] used the HMM states of clustered triphones as the output target classes of DNNs. In this work, we show that an accurate modeling of context dependency is achieved by using the WFST-based structure instead of triphone target classes. The proposed method can be assumed to be a conditional neural field as proposed in [9, 10] where the target label spaces are further expanded by using WFST-based representations. To speed-up the training, we employed DNNs with bottleneck shapes, and the hidden activation of the bottleneck layer is used as feature vectors of a WFST structured classification system. Since it has been reported that the pretraining techniques developed for DNNs are also valid for extracting efficient bottleneck features [11], we can realize an effective combination of these two methods by employing these techniques.

2. WFST-based structured deep neural networks

In this section, we derive an integrated model, called WFST-DNN, by generalizing the WFST decoding processes of DNN-based speech recognition systems.

In WFST decoders, the most plausible word sequence $\hat{\boldsymbol{\ell}}$, given the observation vector sequence $\mathbf{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_T\}$, is obtained by enumerating the output symbols $\boldsymbol{O}[\hat{\mathbf{a}}]$ corresponding to the most plausible WFST arc sequence $\hat{\mathbf{a}}$, as follows:

$$\hat{\boldsymbol{\ell}} = \boldsymbol{O}[\hat{\mathbf{a}}] \text{ where } \hat{\mathbf{a}} = \underset{\mathbf{a} \in \boldsymbol{\mathcal{D}}}{\operatorname{argmax}} P(\mathbf{a}|\mathbf{X}).$$

Here, $\boldsymbol{\mathcal{D}}$ denotes a decoding network that is treated as a set of possible arc sequences.

Typically each arc a_i in an arc sequence $\mathbf{a} = \{a_1, a_2, \cdots, a_i, \cdots\}$ has annotations of the input HMM-state symbol $I[a_i]$, output word symbol $O[a_i]$, time-stamp $T[a_i]$, and arc transition cost $W[a_i]$. Here, language, pronunciation, and HMM transition model probabilities are embedded in the arc

transition costs $W[a_i]$. By using this notation, the arc posterior probability $P(\mathbf{a}|\mathbf{X})$ can be written by using the sum of the arc-wise cost function $\omega^{\texttt{AM}}(a_i;\mathbf{X})$, which is defined as the sum of acoustic cost $g(a_i;\mathbf{X})$ and arc transition cost $W[a_i]$, as follows:

$$\begin{aligned} P(\mathbf{a}|\mathbf{X}) &\propto \exp\left\{\sum_i -\omega(a_i;\mathbf{X})\right\}, \\ \omega^{\texttt{AM}}(a_i;\mathbf{X}) &= g(a_i) + W[a_i], \end{aligned} \tag{1}$$

where the acoustic cost function $g(a_i;\mathbf{X})$ is defined as follows:

$$g(a_i;\mathbf{X}) = \begin{cases} -\log P(\boldsymbol{x}_{T[a_i]}|s = I[a_i]) & I[a_i] \neq \epsilon, \\ 0 & \text{otherwise.} \end{cases}$$

By introducing DNN-based acoustic modeling, $\log P(\boldsymbol{x}_{T[a_i]}|s = I[a_i])$ in the above equation can be expressed as follows:

$$\begin{aligned} &\log P(\boldsymbol{x}_{T[a_i]}|s = I[a_i]) \\ &\quad = \log P(s = I[a_i]|\boldsymbol{x}_{T[a_i]}) - \log P(s = I[a_i]) + C \\ &\quad = \sum_d w_{I[a_i],d} h_d(\boldsymbol{x}_{T[a_i]}) + b'_{I[a_i]} - \log P(s = I[a_i]) + C, \end{aligned}$$

where C is a constant, $w_{I[a_i],d}$ is a connection weight between the output unit corresponding to the class $I[a_i]$ and the d-th hidden unit in the last hidden layer, $b'_{I[a_i]}$ is a bias term of the output unit, and $h_d(\boldsymbol{x}_{T[a_i]})$ is the output value of d-th hidden unit in the last hidden layer. By substituting this expression and $b_{I[a_i]} \stackrel{\text{def}}{=} b'_{I[a_i]} - \log P(s = I[a_i]$ into Eq. (1), the arc transition cost function can be denoted as follows:

$$\omega^{\texttt{DNN}}(a_i;\mathbf{X}) = \boldsymbol{\lambda}(a_i)^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{x}_{T[a_i]}), \tag{2}$$

where feature vectors $\boldsymbol{\phi}(\boldsymbol{x}_{T[a_i]})$ and model parameters $\boldsymbol{\lambda}_{I[a_u]}$ are defined as follows:

$$\boldsymbol{\phi}(\boldsymbol{x}_{T[a_i]}) = \begin{cases} [1, 1, h_1(\boldsymbol{x}_{T[a_i]}), h_2(\boldsymbol{x}_{T[a_i]}), \cdots]^{\mathsf{T}} & I[a_i] \neq \epsilon, \\ [1, 0, 0, 0, \cdots]^{\mathsf{T}} & \text{otherwise.} \end{cases}$$

$$\boldsymbol{\lambda}(a_i) = \begin{cases} [W[a_i], b'_{I[a_i]}, w_{I[a_i],1}, w_{I[a_i],2}, \cdots]^{\mathsf{T}} & I[a_i] \neq \epsilon, \\ [W[a_i], 0, 0, 0, \cdots]^{\mathsf{T}} & \text{otherwise.} \end{cases} \tag{3}$$

From this equation, it is shown that the DNN-based speech recognition is performing structured linear classification where the feature function is defined by using the hidden activation of the DNNs, and the parameters are tied corresponding to the input symbol of the arc.

By introducing the idea of WFST-based structured classifiers, the parameters in the above equation no longer need to be tied by the HMM states. Instead of using the HMM state, the WFST-DNN classifier introduces arc identifier variable $N[a_i]$, and uses it to define the parameter vector as follows:

$$\boldsymbol{\lambda}(a_i) = \boldsymbol{\theta}_{N[a_i]}. \tag{4}$$

By directly optimizing $\boldsymbol{\theta}_{N[a_i]}$ under specific criteria, we can obtain an effective structured linear classifier on the feature vector $\boldsymbol{\phi}$ computed by DNNs.

Figure 1 is a schematic overview of a WFST-DNN model. We can consider WFST-DNN models as neural networks that inherit the deep structure of DNNs and a large output layer corresponding to each arc of the WFSTs.

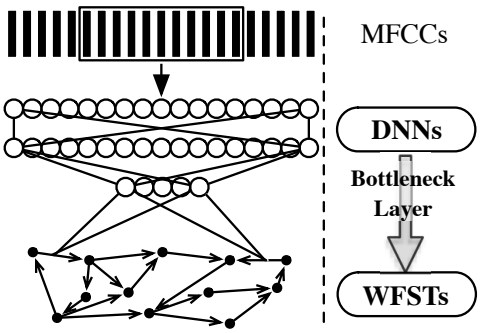

Figure 1: Schematic overview of WFST-DNN model.

3. Two-stage training approach

In this work, we employed a two-stage training approach that calculates lattices by using normal DNNs in the first stage. Then, in the second stage, DNNs used to calculate the lattices are expanded into the equivalent WFST-DNN classifiers, and the WFST-DNNs are optimized further.

3.1. Training of DNNs

Firstly, the proposed method performs the conventional procedures to construct DNN-based acoustic models. Here, to prevent overfitting and computational inefficiency in the subsequent processing of the WFST-DNN classifiers, the DNNs are designed to have a bottleneck layer that consists of a rather smaller number of hidden units. For example, in the following experiments, we used 2048 hidden units for each layer, 512 hidden units for the bottleneck layer, which is located between the output layer and the hidden layers, and 144 (or 834) output units. Each layer, including the bottleneck layer, is pretrained as a restricted Boltzmann machine, and then concatenated DNNs are trained by using the back propagation algorithm. The decoder lattices are then generated by using the trained DNNs.

3.2. Retraining as WFST-DNNs

In the second stage, the parameters of the DNNs used to calculate the lattices are untied and converted into the equivalent parameters of WFST-DNNs by using Eqs. (3) and (4). By using the tied version of the parameters as an initial value of the WFST-DNN systems, we can consistently use pre-computed lattices without crucial gaps in the competing sequences.

Hereafter, we denote a set of parameters that is considered to be in the second stage as $\Theta \stackrel{\text{def}}{=} \{\boldsymbol{\theta}_j|\forall j\}$ where j is an index variable that denote one of arc identifiers. Further, we assume that each observation sequence $\mathbf{X}^{(n)}$ in the given training data set $\{\mathbf{X}^{(n)}|\forall n\}$ has a corresponding *correct* reference arc sequence $\mathbf{a}^{(n)}$. To retrain the parameters Θ, we adopted the minimum transition error training methods used for WFST-based structured classifiers [12]. In this paper, as in [12], we consider the two objective functions for MTE training. The first objective function is a boosted MMI function [13] combined with a minimum transition error (MTE) measure, as follows:

$$F_\sigma^{\texttt{bMMI}}(\Theta) = \sum_n \log \frac{\exp\left\{-\Omega\left(\mathbf{X}^{(n)}, \mathbf{a}^{(n)}; \Theta\right)\right\}}{\sum_{\mathbf{a}'} \exp\left\{-\Omega\left(\mathbf{X}^{(n)}, \mathbf{a}'; \Theta\right) + \sigma E(\mathbf{a}^{(n)}, \mathbf{a}')\right\}}, \tag{5}$$

where $E(\mathbf{a}^{(n)}, \mathbf{a}')$ is the transition error count function that counts the number of frames processed by an incorrect arc.

$\Omega(\mathbf{X}, \mathbf{a}; \Theta)$ is the total cost function defined as follows:

$$\Omega(\mathbf{X}, \mathbf{a}; \Theta) \stackrel{\text{def}}{=} \sum_i \omega(\mathbf{X}, a_i). \tag{6}$$

The bMMI training acts so that the appearance of erroneous arc sequences $\mathbf{a}'$, which yield high $E(\mathbf{a}^{(n)}, \mathbf{a}')$, is strongly suppressed. In the above formulation, we assumed that the reference arc alignment was fixed to ensure convexity in the second stage of the training; however, we can also use lattice-based marginalization in the numerator part of the objective function.

To minimize the error measure E directly, the differenced MMI (dMMI) method [14] exploits the following identity:

$$\left\langle -E(\mathbf{a}^{(n)}, \mathbf{a}') \right\rangle_{P(\mathbf{a}'|\mathbf{X}, \boldsymbol{\Theta})} = \frac{\partial}{\partial \sigma} F_{\sigma}^{\texttt{bMMI}}(\Theta). \tag{7}$$

By computing the above partial derivative numerically, the following dMMI method is obtained.

$$F_{\sigma_1, \sigma_2}^{\texttt{dMMI}}(\Theta) = \frac{F_{\sigma_2}^{\texttt{bMMI}}(\Theta) - F_{\sigma_1}^{\texttt{bMMI}}(\Theta)}{\sigma_2 - \sigma_1}. \tag{8}$$

The dMMI objective function is closely related to the sequence-level minimum Bayes risk training of neural networks [15]. If we use a $\sigma_1 \to +0, \sigma_2 \to -0$ setting the dMMI training converges to the minimum Bayes risk training; however, the target output class of the proposed method is expanded to WFST arcs. Furthermore, if we set $\sigma_1 = \sigma$ and $\sigma_2 \to -\infty$, the bMMI objective function is recovered from the dMMI objective function. Although dMMI can directly minimize the number of transition errors, the objective function is not generally convex.

4. Experiments

To validate the efficiency of the proposed method, we conducted continuous phoneme recognition experiments based on the TIMIT dataset. We used 11-frames of Mel-frequency cepstral coefficients (MFCC)-based 39 dimensional features computed for each 10 ms as DNN input vectors. As we mentioned above, we used 2048 hidden units for each hidden layer, and 512 hidden units for the bottleneck layer. The activation functions of the hidden and bottleneck layers were fixed to the sigmoid function. The optimization of the DNNs (the first stage) was performed in a similar way to that described in [1]. The slight difference with [1] was that we used 144 HMM states corresponding to 48 reduced phoneme sets as an output class, we reduced the learning rate when the frame error rate of the development dataset was increased, and we did not revert to the previous parameters when the validation error increased. Regardless of these small changes, we confirmed that the DNNs' efficiency was also valid under our experimental conditions. The phoneme error rates of our baseline DNN systems are listed in Table 1.

To obtain the arc-identifiers $N[a_i]$, we numbered all the arcs in the decoding WFST. In the experiments, the decoding WFST $\boldsymbol{\mathcal{D}}$ was constructed as $\mathrm{Opt}(\boldsymbol{\mathcal{H}} \circ \boldsymbol{\mathcal{G}})$ where $\boldsymbol{\mathcal{H}}$ was the HMM state sequence network, $\boldsymbol{\mathcal{G}}$ was the phoneme bigram model estimated by using maximum-likelihood training, $\circ$ denoted the composition operation, and Opt denoted an optimization operator of WFSTs. We should note that we did not include the factorization operation in Opt to ensure the equivalent conversion between the conventional arc score (Eq. (1)) and the arc-based score (Eq. (2)). There were 1771 arcs (arc identifiers) in the network $\boldsymbol{\mathcal{D}}$.

The second stage of the WFST-DNN optimization was performed in a batch manner by using the Rprop method [16]. The initial step size of the Rprop was set at a small value (10^{-4}) to respect the solution of the first stage, and the validity of lattices generated by using baseline DNNs. The lattice smoothing factor for optimization was chosen by using the development dataset. In the following experiments, we selected two numbers of the hidden layers, $L = 2$ and $L = 5$, since $L = 2$ was considered sufficiently small while preserving *deepness*, and $L = 5$ performed best in the baseline DNN experiments.

Table 1: Performance of DNNs without a bottleneck (L denotes the number of hidden layers used, and the phoneme error rates (PERs) are displayed as a percentage

L	1	2	3	4	5	6
PER (test)	24.5	23.7	22.9	23.1	**22.7**	22.9
PER (dev.)	23.4	22.5	21.7	21.8	**21.6**	21.7

Table 2: Phoneme error rates (PERs) of the WFST-DNN systems (core test set).

method	σ	PER [%] $L = 2$	PER [%] $L = 5$
ML-HMM	–	32.1	
WFST-CRF [12]	$(2.0, -2.0)$	28.8	
DNN	–	23.7	22.7
BN-DNN	–	23.7	23.1
WFST-DNN bMMI	0.0	23.7	22.5
WFST-DNN bMMI	1.0	23.2	22.2
WFST-DNN bMMI	2.0	22.7	22.5
WFST-DNN bMMI	4.0	22.8	22.6
WFST-DNN dMMI	$(2^{-8}, -2^{-8})$	22.9	**21.9**
WFST-DNN dMMI	$(1.0, -1.0)$	**22.6**	**21.9**
WFST-DNN dMMI	$(2.0, -2.0)$	22.9	22.6
WFST-DNN dMMI	$(4.0, -4.0)$	23.9	23.3

The phoneme error rates of the conventional maximum likelihood HMMs (ML-HMM), conventional WFST-based classifier without DNNs (WFST-CRF), DNN-based acoustic models (DNN), DNNs with an additional bottleneck (BN) layer (BN-DNN), and WFST-DNN systems (WFST-DNN) are listed in Tables 2 and 3. We confirmed that the proposed WFST-based classification can reduce the phoneme errors even from the strong baseline constructed with the DNNs. Even though the relative error reduction rates were around 4%, the improvement from the strong DNN baselines was considered to be important.

In relatively shallow networks ($L = 2$), the introduction of a bottleneck layer did not degrade the performance. This might be attributed to the fact that the number of layers was increasing even though the number of hidden units in the bottleneck layer was small. However, with the deep network, we observed that introducing a small bottleneck layer degraded the performance. This suggests that introducing a bottleneck eliminated information of features during the RBM-based pretraining step; therefore, the following fine tuning converged to a slightly poor local optimum. Even in that case, by introducing WFST-DNN architectures, the total performance was superior to the conventional DNN acoustic models.

We confirmed that the importance of using appropriate hyper-parameters σ, σ_1, and σ_2. By setting these parameters at radical values, the obtained WFST-DNNs became inferior to the original DNN acoustic models. However, we also confirmed that these values can be effectively tuned by using the development dataset. We observed that the best setting in terms of the error rates of the development set also achieved the better performance in the core test set.

Table 4 shows the performance of the WFST-DNN models used with triphone WFSTs. In this comparison, we constructed

Table 3: Phoneme error rates (PERs) of the WFST-DNN systems (development set).

method	σ	PER [%] $L = 2$	PER [%] $L = 5$
ML-HMM	–	30.8	
WFST-CRF [12]	$(2.0, -2.0)$	28.1	
DNN	–	22.5	21.6
BN-DNN	–	22.2	21.8
WFST-DNN bMMI	0.0	21.5	21.3
WFST-DNN bMMI	1.0	21.2	21.0
WFST-DNN bMMI	2.0	21.0	20.8
WFST-DNN bMMI	4.0	21.2	21.0
WFST-DNN dMMI	$(2^{-8}, -2^{-8})$	21.0	**20.7**
WFST-DNN dMMI	$(1.0, -1.0)$	**20.9**	20.8
WFST-DNN dMMI	$(2.0, -2.0)$	21.1	21.0
WFST-DNN dMMI	$(4.0, -4.0)$	22.1	21.7

Table 4: Phoneme error rates (PERs) of deep neural networks (DNNs), context dependent deep neural networks (CD-DNNs), and the proposed methods (WFST-DNN).

	DNN mono	WFST-DNN mono	CD-DNN tri.	WFST-DNN tri.
PER	22.7	21.9	21.9	**21.1**

a decoding network as $\mathrm{Opt}(\boldsymbol{\mathcal{H}} \circ \boldsymbol{\mathcal{C}} \circ \boldsymbol{\mathcal{G}})$ where $\boldsymbol{\mathcal{C}}$ was a context-dependency WFST. There were 3436 arcs in the triphone decoding network. The hyper-parameter we used for the WFST-DNNs was taken from the best setting of the previous experiments. Specifically, we used dMMI training with $L = 5, \sigma_1 = 1, \sigma_2 = -1$ for the WFST-DNNs. Here, we observed that the WFST-based expansion of DNNs worked efficiently. Further, we observed that monophone WFST-DNNs were comparable to triphone the CD-DNNs even though the contextual information leveraged in monophone WFST-DNN models is basically biphone information introduced by the WFST arcs corresponding to the phoneme bigram models. This suggests that the use of expanded label representation is more effective than the use of the conventional context clustering method based on the maximum likelihood principle.

5. Conclusions

This paper proposed a method for integrating deep neural networks (DNNs) with a structured classification method based on weighted finite-state transducers (WFSTs). To take computational advantage of both methods, we employed a two-stage training approach in which baseline DNNs are estimated in the first stage and the last layer of the expanded DNNs, called WFST-DNNs, and WFST weights are jointly estimated in the second stage. To train the WFST-DNNs, we employed minimum transition error (MTE) training methods. By evaluating the WFST-DNNs by using continuous phoneme recognition experiments, we confirmed that MTE training methods were efficient for WFST-DNN training.

Future work will includes the joint optimization of overall systems. Even though the use of a two-stage approach is important for computational efficiency, joint optimization would also be important because it can optimize the bottleneck outputs. This direction can be achieved by scaling up the method proposed in [15]; however, scaling up the method is a challenge because the number of target class increases exponentially in large-vocabulary cases. Furthermore, the use of improved input representations is also considered to be important. Recently, it was shown that the use of filterbank outputs is more efficient than the use of MFCC features [4]. Evaluations of large-vocabulary continuous speech recognition tasks are also considered to be important. Since the second stage of training can be efficiently parallelized, these evaluations can be achieved straightforwardly with the proposed method.

6. Acknowledgments

We thank Dr. Frank Seide of at Microsoft Research Asia for valuable discussions on deep neural networks.

7. References

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

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

[3] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

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

[5] G. Zweig and P. Nguyen, "A segmental CRF approach to large vocabulary continuous speech recognition," in *Proc. IEEE ASRU*, 2009, pp. 152–157.

[6] A. Gunawardana, M. Mahajan, A. Acero, and J. C. Platt, "Hidden conditional random fields for phone classification," in *Proc. INTERSPEECH*, 2005.

[7] S. Watanabe, T. Hori, and A. Nakamura, "Large vocabulary continuous speech recognition using WFST-based linear classifier for structured data," in *Proc. INTERSPEECH*, Aug. 2010, pp. 346–349.

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

[9] J. Peng, L. Bo, and J. Xu, "Conditional neural fields," *Advances in Neural Information Processing Systems*, 2009.

[10] Y. Fujii, K. Yamamoto, and S. Nakagawa, "Deep-hidden conditional neural fields for continuous phoneme speech recognition," in *Proc. International Workshop of Statistical Machine Learning for Speech Processing (IWSML)*, 2012.

[11] D. Yu and M. L. Seltzer, "Improved bottleneck features using pretrained deep neural network," in *Proc. INTERSPEECH*, 2011, pp. 237–240.

[12] Y. Kubo, S. Watanabe, and A. Nakamura, "Decoding network optimization using minimum transition error training," in *Proc. ICASSP*, 2012, pp. 4197–4200.

[13] 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.

[14] E. McDermott, S. Watanabe, and A. Nakamura, "Discriminative training based on an integrated view of MPE and MMI in margin and error space," in *Proc. ICASSP*, 2010.

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

[16] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The Rprop algorithm," in *Proc. IEEE ICNN*, 1993, pp. 586–591.