IEICE TRANS. INF. & SYST., VOL.E95–D, NO.8 AUGUST 2012

PAPER

# Hidden Conditional Neural Fields for Continuous Phoneme Speech Recognition

**Yasuhisa FUJII**[†a)], **Kazumasa YAMAMOTO**[†], *Members*, *and* **Seiichi NAKAGAWA**[†], *Fellow*

**SUMMARY** In this paper, we propose Hidden Conditional Neural Fields (HCNF) for continuous phoneme speech recognition, which are a combination of Hidden Conditional Random Fields (HCRF) and a Multi-Layer Perceptron (MLP), and inherit their merits, namely, the discriminative property for sequences from HCRF and the ability to extract non-linear features from an MLP. HCNF can incorporate many types of features from which non-linear features can be extracted, and is trained by sequential criteria. We first present the formulation of HCNF and then examine three methods to further improve automatic speech recognition using HCNF, which is an objective function that explicitly considers training errors, provides a hierarchical tandem-style feature and includes a deep non-linear feature extractor for the observation function. We show that HCNF can be trained realistically without any initial model and outperforms HCRF and the triphone hidden Markov model trained by the minimum phone error (MPE) manner using experimental results for continuous English phoneme recognition on the TIMIT core test set and Japanese phoneme recognition on the IPA 100 test set.

*key words: hidden conditional neural fields, hidden conditional random fields, hidden Markov model, speech recognition, deep learning*

## 1. Introduction

Current automatic speech recognition (ASR) systems employ a Hidden Markov Model (HMM) together with a Gaussian Mixture Model (GMM) as the emission probability for an acoustic model. However, the HMM has two major drawbacks for use as an acoustic model. First, it relies on a strong independency assumption whereby frames are independent given a state, and thus lacks the ability to deal with features that straddle several frames. Second, because the HMM is a generative model, it is not suitable for discriminating sequences. To solve the former problem, features that can deal with phenomena straddling multiple frames have been developed, such as the delta coefficient [1], segmental statistics [2], and modulation spectrum [3]. For the latter problem, discriminative training methods such as the minimum phone error (MPE) have been investigated [4]. Combining a model that can consider long range features and has high discriminative power with HMM has also been conducted to tackle the above problems. For example, the Tandem system extracts features using a Multi Layer Perceptron (MLP) and uses them as an input for the HMM [5]. Conditional Random Fields (CRF) [6] have also been used instead of MLP [7].

The Hidden Conditional Random Fields (HCRF) is a very promising approach for modeling speech [8]–[10]. HCRF overcomes the above two major drawbacks of HMM while preserving its merits such as efficient algorithms that include a forward-backward algorithm and Viterbi decoding. Although HCRF is promising, it has the drawback that it cannot accommodate non-linearity features, which is crucial for speech recognition, because it computes the score of a hypothesis by summing up linearly weighted feature values. There is an attempt to expand feature vectors explicitly [11]. However, it will be difficult to use the approach when the number of features increases.

Peng et al. proposed Conditional Neural Fields (CNF) which can consider non-linearity among its features by incorporating the gate function of an MLP into the CRF [12]. In this study, we propose *Hidden Conditional Neural Fields (HCNF)* which can consider non-linearities between features by introducing the gate function of an MLP into the HCRF. In an analogy with HCRF, HCNF can incorporate many types of features and be trained using sequential criteria. Prabhavalkar et al. extended CRF by incorporating a trainable gate layer as we did in this paper [13]. Although it seems that our work is close to their work, they extended CRF to incorporate a gate function, and thus, it was closer to CNF than to our work. Using a fixed state alignment instead regarding state as a hidden variable works practically well for MLP-like classifiers such as CNF. Even though they practically work well, realignment using the MLP-like classifiers after the training with the fixed alignment generally provides better results. This means that solving alignment problem together would provide better results. HCNF can solve the alignment in the model and does not need realignment as MLP-like classifiers do. This is a theoretical advantage of HCNF over conventional CNF. Kingsbury introduced a sequential training criterion for training MLP in [14]. Since HCNF can be considered as a combination of HCRF and MLP, and the MLP is trained by a sequential criterion, HCNF seems to be similar to [14]. The main difference between our work and [14] is that in HCNF all parameters are trained together without the distinction of acoustic and linguistic features while in [14] the acoustic model by the MLP and the language model are trained separately. Another merit of the HCNF is that all information is expressed as a feature and it is not restricted only to acoustic and language models.

In addition to providing the basic definition of HCNF

Manuscript received January 10, 2012.
Manuscript revised April 20, 2012.

[†]The authors are with the Department of Information and Computer Sciences Toyohashi University of Technology, Toyohashi-shi, 441–8580 Japan.
a) E-mail: fujii@slp.ics.tut.ac.jp

DOI: 10.1587/transinf.E95.D.2094

in this study, we propose three facilities to further improve ASR using HCNF, namely, provide an objective function which can consider training error more explicitly, a hierarchical tandem-style feature and a deep non-linear feature extractor for the observation function.

The first facility is to provide an objective function that can consider training error more explicitly than straightforward posterior maximization. HCNF can be trained based on posterior probability maximization, which corresponds to the maximum mutual information (MMI) criterion in the context of an HMM based acoustic training model. However, as is well known, improving the posterior probability does not necessarily decrease the final error rates which depend on the tasks such as Word Error Rates (WER) and Phoneme Error Rates (PER). Therefore, a training criterion that considers the final error rates is preferable to a straightforward posterior probability maximization. Although considering the same error criterion at both training phase and test phase would be considered to provide the best results, it has been shown in [15] that test set word error rates are not affected so much by how to compute errors if the granularity of computing errors at the training phase is higher than word level. Based on the result, in this paper, we consider training errors at state level since it is easy to compute by HCNF. Among several objective functions which can consider training error explicitly, the Boosted MMI is a promising criterion due to its effectiveness and easiness of implementation [16]. The same criterion for log-linear models (HCRF) is discussed in [17]. Although these approaches seem effective for HCNF training, they cannot be applied to HCNF directly because the state sequences are not observable (i.e., they are hidden) in HCNF. In this study, we propose the Hidden Boosted MMI (HB-MMI) as a new training criterion for HCNF. The HB-MMI considers training errors more directly than posterior maximization even if the state sequences are not known (fixed). Instead of taking forced alignment and using fixed alignment, we compute the expected error counts from the posteriors of reference state sequences and all hypotheses state sequences.

The second facility is the hierarchical tandem-style feature using HCNF. The posterior features are robust to variability in context, speaker, and noise compared with raw acoustic features such as Mel-frequency cepstrum coefficients (MFCC) and perceptual linear prediction (PLP), and accordingly have shown improved recognition accuracy in several studies [5], [7], [18]. In [18], the hierarchical phoneme posterior feature which is extracted using the first MLP and is used as a feature for the second MLP is explored and showed its effectiveness. In this study, we make use of the idea from [18], except we use the HCNF instead of MLP. By using HCNF, we do not need fixed state alignment which is required if MLP is used.

The final facility is the deep non-linear feature extractor for the observation function of HCNF. Recently, deep densely connected feed forward neural networks have achieved remarkable success [19]. In this paper, we propose to use an observation function with a deep structure in HCNF. The proposed deep observation function enables the use of the deep feed forward neural networks in HCNF.

We used the TIMIT corpus to examine the effectiveness of our proposed method because it offers a good test bed for studying algorithmic improvements [20]. In addition to the TIMIT corpus, we used the ASJ+JNAS corpus which is about 11 times larger than the TIMIT corpus to show the scalability of HCNF. We show that HCNF can be trained realistically without any initial model and outperforms HCRF and the triphone HMM trained by the MPE manner using experimental results for continuous phoneme recognition in these corpora. This paper is organized as follows. In the next section, HCRF for ASR is described. Section 3 introduces HCNF. The three proposed methods, the HB-MMI, the hierarchical state posterior feature and the deep objective function are then explained in Sects. 4, 5 and 6, respectively. In Sect. 7, the training methods for HCRF and HCNF are explained. The experimental setup and results are shown in Sect. 8. Finally, Sect. 9 presents our conclusions and outlines possible future work.

## 2. Hidden Conditional Random Fields

### 2.1 Formulation

Given an observation sequence $X = (x_1, x_2, \dots, x_T)$, HCRF computes the score of a label sequence $Y = (y_1, y_2, \dots, y_T)$ as follows:

$$P(Y|X) = \frac{\sum_S \exp(\kappa(\Phi_r(X, Y, S) + \Psi_r(X, Y, S)))}{Z(X)}, \tag{1}$$

where $\kappa$ is a state-flattening coefficient [21]. $Z(X)$ is a partition function and computed as follows:

$$Z(X) = \sum_{Y'} \sum_{S} \exp(\kappa(\Phi_r(X, Y', S) + \Psi_r(X, Y', S))), \tag{2}$$

$\Phi_r(X, Y, S)$ and $\Psi_r(X, Y, S)$ are an observation function and a transition function, respectively. $r$ is used to express that these are feature functions of HCRF. In Sect. 3, $n$ is used to express feature functions of HCNF. The feature functions of HCRF are defined as follows:

$$\Phi_r(X, Y, S) = \sum_k w_k \sum_t \phi_k(X, Y, S, t), \tag{3}$$

$$\Psi_r(X, Y, S) = \sum_j u_j \sum_t \psi_j(X, Y, S, t, t-1). \tag{4}$$

$S = (s_1, s_2, \dots, s_T)$ is a hidden variable sequence that represents a state sequence. $\phi_k(X, Y, S, t)$ and $\psi_j(X, Y, S, t, t-1)$ mean a raw observation feature extracted at frame $t$ and a transition feature extracted at frame $t$ and $t-1$, respectively. The corresponding weights are $w_k$ and $u_j$. Efficient algorithms such as the forward-backward and Viterbi algorithms can be adapted to HCRF because the feature functions used in HCRF are restricted to the current frame in the observation function, and current and previous frames in the transition function.

## 2.2 Objective Function

Given training data $D = \{X^i, Y^i\}, i = 0, \ldots, N$, the objective function of HCRF is set as follows:

$$\ell(\lambda; D) = -\sum_i \log P(Y^i|X^i). \tag{5}$$

$\lambda = \{w_k, u_j; k, j = 1, 2, \ldots\}$ which minimizes $\ell(\lambda; D)$ can be found using gradient based methods such as L-BFGS and Stochastic Gradient Descent (SGD). The partial derivatives of Eq. (5) with respect to $w_k$ can be computed as follows:

$$\begin{aligned}\frac{\partial \ell(\lambda; D)}{\partial w_k} =& -\kappa \sum_i E\left[\sum_t \phi_k(X^i, Y^i, S, t)\right]_{S|Y^i,X^i} \\ &+ \kappa \sum_i E\left[\sum_t \phi_k(X^i, Y, S, t)\right]_{Y,S|X^i},\end{aligned} \tag{6}$$

where $E[]_X$ means the expectation by $X$. The partial derivatives with respect to $u_j$ can be computed as follows:

$$\begin{aligned}\frac{\partial \ell(\lambda; D)}{\partial u_j} =& -\kappa \sum_i E\left[\sum_t \psi_j(X^i, Y^i, S, t, t-1)\right]_{S|Y^i,X^i} \\ &+ \kappa \sum_i E\left[\sum_t \psi_j(X^i, Y, S, t, t-1)\right]_{Y,S|X^i},\end{aligned} \tag{7}$$

In HCRF training, all parameters are trained together without distinguishing acoustic and linguistic features, unlike traditional HMM and N-gram training.

## 2.3 Decoding

Reference [10] used the N-best decoding to find the sequence that maximizes Eq. (1). Although the method yielded a reasonable result, it is difficult to adapt it directly to large vocabulary continuous speech recognition (LVCSR). Therefore, we decided to use the Viterbi algorithm for the decoding. This means that the hidden variable $S$ is not marginalized out in the decoding. The feature functions used in HCRF are restricted to the current frame in the observation function, and the current and previous frames in the transition function. In such case, Viterbi algorithm can be used to find the best state sequence in analogy with HMM.

## 3. Hidden Conditional Neural Fields

### 3.1 Formulation

HCNF becomes an extension of HCRF by introducing a gate function into it. Figure 1 shows the structures of HCRF and HCNF. In HCRF, the score of a pair of current state and label, $(s_i, y_i)$, is computed by summing up linearly weighted feature values. On the other hand, HCNF computes the score of the pair by summing up linearly weighted gate values, which are obtained by applying a non-linear function to the score computed by summing up linearly weighted feature values. Thus, in HCNF, features are implicitly extracted by a non-linear function at the gates level and their gates are used to compute scores of label and state sequences. HCNF computes a probability $P(Y|X)$ as follows:

$$P(Y|X) = \frac{\sum_S \exp(\kappa(\Phi_n(X, Y, S) + \Psi_n(X, Y, S)))}{Z(X)}, \tag{8}$$

where $Z(X)$ is a partition function computed as:

$$Z(X) = \sum_{Y'} \sum_S \exp(\kappa(\Phi_n(X, Y', S) + \Psi_n(X, Y', S))), \tag{9}$$

$\Phi_n(X, Y, S)$ and $\Psi_n(X, Y, S)$ are an observation function and a transition function, and are given by:

$$\Phi_n(X, Y, S) = \sum_t \sum_g^K w_{y_t,s_t,g} h(\theta^T_{y_t,s_t,g} \phi(X, Y, S, t)), \tag{10}$$

$$\Psi_n(X, Y, S) = \sum_j u_j \sum_t \psi_j(X, Y, S, t, t-1), \tag{11}$$

where $\psi_j(X, Y, S, t, t-1)$ is a transition feature extracted at frame $t$ and $t-1$, and $u_j$ is a corresponding weight. $w_{y,s,g}$ is a specific weight to the triple $y$, $s$, $g$. $\phi(X, Y, S, t)$ is a vector representation of features such as the MFCCs, and $\theta_{y,s,g}$ is the corresponding weight vector specific to the triple $y$, $s$, and $g$[†]. $h(x)$ is a gate function defined as:

$$h(x) = \frac{c}{1 + \exp(-\alpha(x - \beta))} - d, \tag{12}$$

where $c$ and $d$ are terms to change the range of values, and $\alpha$ and $\beta$ are terms to control the shape of the gate function. In HCNF, the observation function $\Phi_n(X, Y, S)$ uses $K$ gate functions through which it considers non-linearities among the features, while the same transition function as in HCRF is used in HCNF. Because the feature functions used in HCNF are restricted to the current frame in the observation function, and the current and previous frames in the transition function, efficient algorithms such as the forward-backward and Viterbi algorithms can also be used in HCNF.

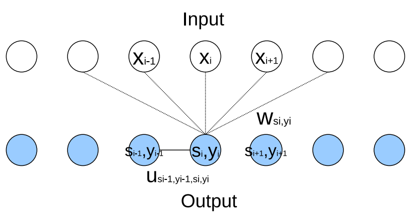
### 3.2 Objective Function

The objective function for HCNF is defined as

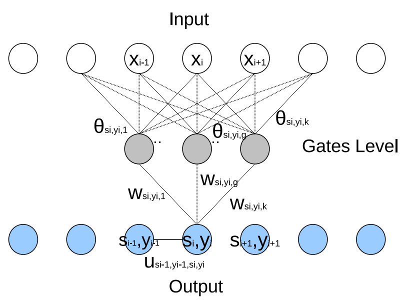
$$\ell(\lambda; D) = -\sum_i \log P(Y^i|X^i). \tag{13}$$

The partial derivatives of (13) with respect to $w_{w,s,g}$, $\theta_{y,s,g}$,

[†]By this definition, the gate functions are dependent on their states, unlike in the original CNF [12]. We used this definition as it was robust in our initial experiments. However, we can obtain the state independent gate functions by setting $\theta_{y_t,s_t,g} \triangleq \theta_g$. The state independent gate functions are used for the deep observation function described in Sect. 6.



(a) HCRF



(b) HCNF

**Fig. 1** The structure of HCRF and HCNF.

and $u_j$ can be computed as follows:

$$\frac{\partial \ell(\lambda; D)}{\partial w_{y,s,g}} = -\kappa \sum_i E\left[\sum_t h(\theta^T_{y,s,g}\phi(X^i, Y^i, S, t))\right]_{S|X^i,Y^i} + \kappa \sum_i E\left[\sum_t h(\theta^T_{y,s,g}\phi(X^i, Y, S, t))\right]_{Y,S|X^i}, \quad (14)$$

$$\frac{\partial \ell(\lambda; D)}{\partial \theta_{y,s,g}} = -\kappa \sum_i E\left[\sum_t w_{y,s,g} \frac{\partial h(\theta^T_{y,s,g}\phi(X^i, Y^i, S, t))}{\partial \theta_{y,s,g}}\right]_{S|X^i,Y^i} + \kappa \sum_i E\left[\sum_t w_{y,s,g} \frac{\partial h(\theta^T_{y,s,g}\phi(X^i, Y, S, t))}{\partial \theta_{y,s,g}}\right]_{Y,S|X^i}. \quad (15)$$

$$\frac{\partial \ell(\lambda; D)}{\partial u_j} = -\kappa \sum_i E\left[\sum_t \psi_j(X^i, Y^i, S, t, t-1)\right]_{S|Y^i,X^i} + \kappa \sum_i E\left[\sum_t \psi_j(X^i, Y, S, t, t-1)\right]_{Y,S|X^i}, \quad (16)$$

The derivative of (12) is computed as:

$$\frac{dh(x)}{dx} = \frac{\alpha}{c}(d + h(x))(c - d - h(x)). \quad (17)$$

In common with HCRF, all parameters are trained together without the distinction of acoustic and linguistic features.

## 3.3 Decoding

We used the Viterbi algorithm to find the most likely sequence of hidden states instead of searching for the most likely output sequence that maximizes Eq. (8).

## 4. Hidden Boosted MMI

In this section, we propose the *Hidden Boosted MMI (HB-MMI)* as a new training criterion for HCNF. HB-MMI considers training errors in a more direct way than posterior maximization even if state sequences are not known (fixed). Instead of taking a forced-alignment and using fixed alignment, we compute expected error counts from the posteriors of reference state sequences and all hypotheses state sequences.

The objective function for HB-MMI is defined as follows:

$$\ell_{HB-MMI}(\lambda; D) = \quad (18)$$
$$-\sum_i \log \frac{\sum_S \exp(\kappa(\Lambda(X, Y, S)))}{\sum_{Y'} \sum_S \exp(\kappa(\Lambda(X, Y, S))) \exp(-b\mathrm{Acc}(S, D^i))},$$
$$\Lambda(X, Y, S) = \Phi_n(X, Y, S) + \Psi_n(X, Y, S), \quad (19)$$

where $\mathrm{Acc}(S, D^i)$ is the expected correct state count of a state sequence $S$ given a reference $D^i = (X^i, Y^i)$. The problem here is that we do not know which state sequence is correct since they are hidden variable. Therefore, we compute expected correct state count between $S$ and $S'$ with posterior probabilities that how likely $S'$ is correct given a training data $D^i$. For that, we first compute the correct state count of state sequence $S$ given the reference state sequence $S'$ as follows:

$$\mathrm{Acc}(S, S') = \sum_t \delta(s_t, s'_t), \quad (20)$$

then, $\mathrm{Acc}(S, D^i)$ is computed as follows:

$$\begin{aligned}\mathrm{Acc}(S, D^i) &= \sum_{S'} P(S'|Y^i, X^i)\mathrm{Acc}(S, S'),\\ &= \sum_{S'} P(S'|Y^i, X^i) \sum_t \delta(s_t, s'_t),\\ &= \sum_t \sum_{S'} P(S'|Y^i, X^i)\delta(s_t, s'_t),\\ &= \sum_t \gamma^i(s_t, t), \end{aligned}\quad (21)$$

where $\gamma^i(s, t)$ is the posterior probability of state $s$ at frame $t$ given reference $D^i$,

$$\gamma^i(s, t) = \sum_{S'} P(S'|Y^i, X^i)\delta(s, s'_t). \quad (22)$$

In other words, $\mathrm{Acc}(S, D^i)$ is defined as the sum of the expected accuracy rates of state $s_t$ at frame $t$. If $\mathrm{Acc}(S, D^i)$ is regarded as a constant, the partial derivative of Eq. (18) can be computed by Eqs. (14)–(16). Since Eq. (21) is computed as the sum of local values at frame $i$, the expectations needed to compute the partial derivatives can easily be computed by merely adding $-b\gamma^i(s_t, t)$ to the score of state $s_t$ at frame $t$, without changing the original algorithm for posterior maximization.

This kind of objective function is closely related to margin maximization [17]. The term $b$ in (18) controls how strongly the margin is imposed in HB-MMI.

## 5. Hierarchical State Posterior Feature

In this section, we propose a method to further improve HCNF based ASR using a hierarchical state posterior feature. The posterior features are robust to variability in context, speaker, and noise compared with the raw acoustic features such as MFCC and PLP, and can provide information for detecting phonemes that tend to be misrecognized as other phonemes given their long-range posterior features. Several studies have improved the recognition accuracy [5], [7], [18].

In [18], a hierarchical phoneme posterior feature extracted by the first MLP and used as a feature for the second MLP was explored and its effectiveness demonstrated. In this study, we make use of the idea in [18], except we use HCNF instead of MLP. That is, the output from the first HCNF is used as an input for the second HCNF. By using HCNF, we obviate the need for time alignment, which is required by MLPs.

We use the state posteriors at each frame as the output from the first HCNF and the input for the second HCNF. This we call the *hierarchical state posterior feature*. The hierarchical posterior feature $\gamma(s, t|X)$ of state $s$ at frame $t$ given an input sequence $X$ is defined as:

$$\gamma(s, t|X) = \sum_{Y'} \sum_{S'} P(Y', S'|X)\delta(s, s'_t). \tag{23}$$

## 6. Deep Observation Function

As shown in Sect. 8.4, the hierarchical posterior feature described in the previous section yields significant improvements. In this section, instead of stacking HCNF, we propose to use a *deep observation function* in HCNF (*Deep-HCNF*). Recently, deep densely connected neural networks have achieved remarkable success [19]. The proposed deep observation function enables the use of deep neural networks in HCNF. To use the deep observation function in HCNF, the following equation is used as the observation function instead of Eq. (10):

$$\Phi_n(X, Y, S) = \sum_t \sum_g^K w_{y_t, s_t, g} G_{L,g}(X, Y, S, t), \tag{24}$$

where $L$ is the number of layers and $G_{l,g}$ is defined recursively as follows:

$$G_{l,g}(X, Y, S, t) = \begin{cases} h(\sum_v^{K_{l-1}} \theta_{l,v} G_{l-1,v}(X, Y, S, t)) & l > 1, \\ h(\theta_{l,g}^T \phi(X, Y, S, t)) & l = 1. \end{cases} \tag{25}$$

When $L = 1$, Eq. (24) reverts to Eq. (10). Partial derivatives of Eqs. (13) and (18) with the deep observation function in terms of $\theta_{l,v}$ can be derived similarly to Eq. (15).

## 7. Training Method

### 7.1 Regularization

Regularization is effective in avoiding over-fitting in HCRF training [9]. Therefore, we regularize the objective functions of both HCRF and HCNF as follows:

$$f(\lambda; D) = l(\lambda; D) + r(\lambda) \tag{26}$$

where $r(\lambda)$ is L1 or L2 regularization:

$$\text{L1:} \qquad r(\lambda) = C\|\lambda\|_1 = C\sum_i |\lambda_i| \tag{27}$$

$$\text{L2:} \qquad r(\lambda) = C\|\lambda\|_2 = \frac{C}{2}\sum_i \lambda_i^2 \tag{28}$$

where $C$ is a trade-off between $l(\lambda; D)$ and the regularization term.

### 7.2 Optimization Algorithm

We adopted SGD to train HCRF and HCNF. SGD is a type of online algorithm and has favorable behavior in that it is not likely to fall into a local optima and has quite fast convergence speed. SGD updates parameters using a gradient $\boldsymbol{g}_t$ computed from a subset of the entire training data in each iteration:

$$\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t - \eta_t \boldsymbol{g}_t, \tag{29}$$

where $\eta_t$ is a learning rate. In this paper, we compute $\boldsymbol{g}_t$ from only one sample (utterance). As an alternative to choosing a sample randomly from the training data, we shuffled the entire training data and used the samples in their order for training in all iterations. Suppose *#iter* means the number of iteration times for the entire training data and *#sample* means the number of samples in training data, then, $\eta_t$ is computed as follows:

$$\eta_t = \alpha \frac{\#sample \cdot \#iter - t}{\#sample \cdot \#iter}. \tag{30}$$

where $\alpha$ determines the value of $\eta_0$. We used FOBOS for SGD with regularization [22].

## 8. Experiments

### 8.1 Setup

We used the TIMIT corpus to examine the effectiveness of our proposed method because it offers a good test bed to study algorithmic improvements [20]. The training set in the TIMIT corpus consists of 3696 utterances by 462 speakers ($\approx 3h$). For the evaluation, we used the core test set consisting of 192 utterances by 24 speakers. We also used the ASJ+JNAS corpus[†], which is about 11 times larger than the TIMIT corpus. The training set in the ASJ+JNAS corpus consists of 20337 utterances by 133 speakers ($\approx 33h$). For evaluation, we used the test sub-set IPA100 consisting of 100 utterances by 23 speakers. We extracted 13 MFCC features together with their deltas and double deltas to form a 39-dimensional observation for the TIMIT corpus. Log energy was used instead of the 0-th MFCC for the ASJ+JNAS corpus. The speech was analyzed using a 25 ms Hamming window with a pre-emphasis coefficient of 0.97 and shifted with a 10 ms fixed frame advance. The acoustic features were normalized to have zero mean and unit variance. For the TIMIT corpus, the 61 TIMIT phonemes were mapped into 48 phonemes for training and further collapsed from 48 to 39 phonemes for evaluation [23]. For the ASJ+JNAS corpus, 43 Japanese phonemes were used. All phonemes were represented as 3 state left-to-right monophone models. We defined four observation functions as given below:

$$\phi^{M1}_{s,d,f}(X,Y,S,t) = \delta(s_t = s)x_{d,t+f}, \tag{31}$$

$$\phi^{M2}_{s,d,f}(X,Y,S,t) = \delta(s_t = s)x^2_{d,t+f}, \tag{32}$$

$$\phi^{Occ}_{s}(X,Y,S,t) = \delta(s_t = s), \tag{33}$$

$$\phi^{Uni}_{y}(X,y,s,t) = \delta(y_t = y), \tag{34}$$

where $x_{d,t}$ is the $d$–th component of $x_t$ and $f$ is used to consider the surrounding frames. In the experiments, we used $-4 \leq f \leq 4$, which means we used the number of observations in the nine frames centered at the current frame. In the experiments considering the hierarchical state posterior feature, a window of $-11 \leq f \leq 11$ (23 frames) was also examined, as in [18]. Moreover, we defined the following transition function:

$$\psi^{Tr}_{s,s'}(X,Y,S,t,t-1) = \delta(s_t = s)\delta(s_{t-1} = s'), \tag{35}$$

$$\psi^{Bi}_{y,y'}(X,y,y',s,s',t,t-1) = \delta(y_t = y)\delta(y_{t-1} = y'). \tag{36}$$

In HCNF, only the M1 and M2 features are treated as observation functions while others are treated as transition functions. All parameters of the HCNF were randomly initialized between −0.5 and 0.5. The state-flattening parameter $\kappa$ was set to 0.1. The number of gates was set to 4 when using delta features and to 16 when not using delta features. When using hierarchical state posterior features, the second HCNF employed 4 gates per state. The parameters of the gate function were set to $\alpha = 0.1$, $\beta = 0.0$, $c = 6.0$, and $d = 3.0$. When hierarchical state posterior features were used, $\alpha = 1.0$ was used. They were tuned on the development set described below and fixed during the training. The parameters for the HCNF were trained using 30 iterations for the TIMIT corpus and 15 iterations for the ASJ+JNAS corpus using SGD with L2 regularization. Regularization parameter $C$ was set to 1.0. All hyper parameters were tuned using a small held-out development set which consisted of 20 utterances randomly extracted from training data.

**Table 1** Phoneme recognition results on the TIMIT core test set [%]. PER means Phoneme Error Rate. Train-PER means PER on training data (48 phonemes).

| Model | PER | Train-PER | #Param |
|---|---|---|---|
| Monophone-HMM (MLE) | 30.0 | 25.0 | |
| Monophone-HMM (MMI) | 28.7 | 17.0 | 366672 |
| Monophone-HMM (MPE) | 28.4 | 18.4 | |
| HCRF (none) | 31.6 | 21.7 | |
| HCRF (L1) | 30.8 | 20.1 | 104928 |
| HCRF (L2) | 30.5 | 20.5 | |
| HCNF (none) | 30.8 | 10.4 | |
| HCNF (L1) | 29.3 | 12.5 | 412656 |
| HCNF (L2) | 28.0 | 16.1 | |
| HCRF [10] | 28.3 | - | - |

For comparison, we conducted recognition experiments using HMMs with the same topology as the HCNFs (monophone). Initially, the HMM with diagonal covariance matrices and a 32 mixture GMM, was trained in an maximum likelihood estimation (MLE) manner, using the hidden Markov model toolkit (HTK). The MLE-HMM model was used to train MMI- and MPE- HMMs with an I-smoothing set to 100, a learning rate parameter of 2 and a scaling factor of 0.2 (similar to the state-flattening coefficient used in HCNFs), and the parameters were updated 10 times. A bigram phone language model was trained from the training corpora. In HCRF and HCNF, the transition function expresses equivalent information to the bigram language model. In addition to the monophone HMMs, we created tied-state triphone HMMs to obtain the best results with HMMs. For the TIMIT experiment, the HMMs consisted of 792 states which had diagonal covariance matrices and an 8 mixture GMM, while for the ASJ+JNAS experiment, they consisted of 2087 states which had diagonal covariance matrices and a 32 mixture GMM. The results using the triphone-based HMMs are shown in Sect. 8.5.

## 8.2 Basic Results

Table 1 shows the phoneme recognition results on the TIMIT core test set. We can see that both regularizations consistently improved the performance of HCRF and HCNF. The results for the HCRFs were inferior to the results of HMMs and significantly worse than the result of PER = 28.2 reported in [10]. This is because of our implementation of HCRF in which mixtures were not used, unlike in [10]. Instead of using mixtures in HCRF, we extended it to HCNF by introducing gate functions. HCNFs clearly outperformed HCRFs and the result showed the effectiveness of incorporating the gate function into HCRF. When using L2 regularization, we obtained the best performance of PER = 28.0. This result was superior to the results of HMMs and comparable with the best of the previous results in the monophone setting. The parameter number of HCNF

[†]http://www.mibel.cs.tsukuba.ac.jp/_090624/jnas/instruct.html

**Table 2** Phoneme recognition results on the IPA100 test set [%].

| Model | PER | Train-PER | #Param |
|---|---|---|---|
| Monophone-HMM (MLE) | 19.0 | 18.7 | 366672 |
| Monophone-HMM (MMI) | 15.8 | 15.6 | |
| Monophone-HMM (MPE) | 15.0 | 13.2 | |
| HCRF (L2) | 16.8 | 16.2 | 104928 |
| HCNF (L2) | 15.7 | 14.3 | 412656 |

**Table 3** Phoneme recognition results using HB-MMI on the TIMIT core test set [%].

(a) with delta features

| $b$ | PER | Train-PER | #Param |
|---|---|---|---|
| 0.0 (MMI) | 28.0 | 16.1 | 412656 |
| 1.0 | 28.5 | 13.8 | |
| 3.0 | 29.2 | 11.9 | |
| 5.0 | 29.1 | 11.4 | |

(b) without delta features

| $b$ | PER | Train-PER | #Param |
|---|---|---|---|
| 0.0 (MMI) | 30.9 | 28.7 | 546480 |
| 1.0 | 29.4 | 26.8 | |
| 3.0 | 27.9 | 24.8 | |
| 5.0 | 28.8 | 23.9 | |

was 412,656 while that of HMM was 366,672 (that of HCRF was 104,928) and they were not significantly different[†].

Table 2 shows the phoneme recognition results for the IPA100 test set. We trained HCRF and HCNF models with only L2 regularization. On the test set, HCNF outperformed HCRF again. However, HCNF was inferior to the HMM trained using the MPE criterion while it was comparable to the HMM trained using the MMI criterion. As shown in Sect. 8.3, HCNF could outperform HMM trained by the MPE criterion using HB-MMI. The results showed the scalability of HCNF since the ASJ+JNAS corpus was about 11 times larger than the TIMIT corpus.

## 8.3 Results of Training with HB-MMI

Table 3 (a) gives the phoneme recognition results on the TIMIT core test set with several values of $b$ in Eq. (18) using the delta features. Since $b$ is a term that determines how strongly the margin is imposed in HB-MMI, the larger $b$ is, the stricter the algorithm in terms of training error. In Table 3 (a), the results for $b = 0$, which corresponds to the MMI criterion, that is, when HB-MMI is not used, are the best. Although we observed a reduced training error (Train-PER) with a larger $b$, it appears to be because of over-fitting.

Table 3 (b) shows the phoneme recognition results on the TIMIT core test set when delta features are not used. In this case, in contrast to when delta features are used, HB-MMI is effective and improves the recognition results. The best result in Table 3 (b) occurs with $b = 3.0$, which is comparable to the best result in Table 3 (a) where delta features are used. This result shows that HCNF is able to extract discriminative features by considering training errors explicitly using HB-MMI even if delta features are not used. Interestingly, the PERs on training data in Table 3 (b) are much higher than the PERs on training data in Table 3 (a). These

**Table 4** Phoneme recognition results with delta features when the number of feature dimensions were reduced using PCA on the TIMIT core test set [%].

| $b$ | PER | Train-PER | #Param |
|---|---|---|---|
| 0.0 (MMI) | 28.5 | 18.3 | 546480 |
| 1.0 | 27.8 | 16.0 | |
| 3.0 | 27.8 | 13.7 | |

results indicate the models trained without delta features are more robust than the models trained with delta features.

Although the models trained without delta features seem robust than the models trained with delta features, it is not yet clear that whether the robustness comes from the smaller number of feature dimensions or the unnecessity of the redundant feature expansion by the delta coefficients. To explore this question, we conducted an additional experiment with delta features where the number of feature dimensions was reduced using Principal Component Analysis (PCA). The number of feature dimensions with the delta features was $702 = 78 \cdot 9$ and the one without the delta features was $234 = 26 \cdot 9$. We computed a PCA matrix for the features with the delta coefficients on the training data and projected the 702 dimensional features into a 234 dimensional space using the matrix. Experimental results when the projection matrix was used for both the training and test data are shown in Table 4. We used $K = 16$ for this experiment and thus the number of parameters in Table 4 is the same as Table 3 (b). The contribution ratio of the reduced space to the original space was 95.5%. Although PER increased slightly on the test set by reducing the number of features, it decreased and outperformed the best result shown in Table 3 (a) by considering errors using HB-MMI. Although the Train-PERs were higher than the ones of Table 3 (a), they were much lower than the ones of Table 3 (b). These results indicate that the robustness of the models trained without delta features mainly comes from the unnecessity of the redundant feature expansion by the delta coefficients and not from the smaller number of features.

To confirm that no effect on HB-MMI with delta features in Table 3 (a) was caused by over-fitting, we conducted experiments on the ASJ+JNAS corpus, which is about 11 times larger than the TIMIT corpus. Table 5 (a) gives the recognition results on this corpus. In the experiment, HB-MMI was clearly effective and improved the recognition results. In addition, HCNF trained by the HB-MMI criterion clearly outperformed the HMM trained by the MPE criterion. From these results, we conclude that HB-MMI would be effective, particularly if over-fitting is not an issue.

Table 5 (b) shows the recognition results on the ASJ+JNAS corpus without delta features. Although HB-MMI was also effective in this case, the result was inferior to the result with delta features. This result indicates that it may be possible to use heuristically extracted features like

[†]HMMs with a GMM of 64 mixtures did not provide any improvements over the HMMs with a GMM of 32 mixtures. Therefore, the number of parameters for HMMs was adequate in this monophone setting.

**Table 5** Phoneme recognition results using HB-MMI on the IPA100 test set [%].

(a) with delta features

| $b$ | PER | Train-PER | #Param |
|---|---|---|---|
| 0.0 (MMI) | 15.7 | 14.3 | 412656 |
| 1.0 | 14.6 | 13.0 | |
| 3.0 | 13.9 | 12.8 | |

(b) without delta features

| $b$ | PER | Train-PER | #Param |
|---|---|---|---|
| 0.0 (MMI) | 19.1 | 19.1 | 546480 |
| 3.0 | 18.0 | 17.3 | |

**Table 6** Phoneme recognition results with the hierarchical state posterior feature on the TIMIT core test set with delta features [%].

(a) with delta features

| Window (stage) | b | PER | Train-PER | #Param |
|---|---|---|---|---|
| 9 (first) | 0.0 | 28.0 | 16.1 (See Table 3 (a)) | 412656 |
| 9 (second) | 0.0 | 26.6 | 13.4 | 1351392 |
| 9 (second) | 3.0 | 26.9 | 12.6 | |
| 23 (second) | 0.0 | 26.2 | 10.9 | 2802912 |
| 23 (second) | 3.0 | 26.9 | 9.4 | |

(b) without delta features

| Window (stage) | b | PER | Train-PER | #Param |
|---|---|---|---|---|
| 9 (first) | 3.0 | 27.9 | 24.8 (See Table 3 (b)) | 546480 |
| 9 (second) | 0.0 | 26.4 | 22.5 | 1485216 |
| 9 (second) | 3.0 | 25.7 | 21.8 | |
| 23 (second) | 0.0 | 26.1 | 18.6 | 2936736 |
| 23 (second) | 3.0 | 25.5 | 17.0 | |
| 23 (second) | 5.0 | 25.3 | 16.6 | |
| 23 (second) | 10.0 | 25.9 | 16.6 | |

the delta features without harming accuracy if over-fitting is not an issue.

## 8.4 Investigation of the Hierarchical State Posterior Feature

We also conducted experiments using the hierarchical state posterior feature, extracted according to the results in Sect. 8.3. The recognition results for $b = 0.0$ in Table 3 (a) were applied when delta features were used in the first HCNF, whereas the recognition results for $b = 3.0$ in Table 3 (b) were applied when not using delta features in the first HCNF. Tables 6 (a) and 6 (b) give the results with and without using delta features in the first HCNF. These tables show that the hierarchical state posterior feature consistently improves recognition results regardless of the use of delta features. The results with a window length of 23 frames are superior to those with a window length of 9. This is consistent with the results in [18]. The best result PER = 25.3 was obtained without using delta features in the first HCNF, with a window length of 23 in the second HCNF, and with the use of HB-MMI and $b = 5.0$ in the second HCNF. The combination of HB-MMI and the hierarchical state posterior feature produced the best results. The results obtained without using delta features in the first HCNF are superior to those obtained with delta features in the first stage HCNF, which could be because models trained without delta features are more robust than those trained with delta features because HCNF can extract useful features without the help of delta features, which act as a type of filter in the modulation spectrum domain.

Experiments on ASJ+JNAS corpus did not provide any improvement with the hierarchical state posterior feature. We need to investigate in which cases the hierarchical state posterior feature is effective.

**Table 7** Phoneme recognition results with the deep observation function on the TIMIT core test set [%].

| #Layer | PER | Train-PER | #Param |
|---|---|---|---|
| 1 | 32.4 | 32.3 | 194040 |
| 2 | 30.8 | 30.1 | 444040 |
| 3 | 29.7 | 28.4 | 694040 |
| 4 | 28.6 | 27.0 | 944040 |
| 5 | 27.8 | 25.7 | 1196540 |
| 10 | 26.0 | 21.2 | 2444040 |
| 15 | 25.5 | 16.8 | 3694040 |

**Table 8** Phoneme recognition results with the single layer observation function on the TIMIT core test set [%].

| #Gate | PER | Train-PER | #Param |
|---|---|---|---|
| 1000 | 32.7 | 31.8 | 383040 |
| 2000 | 32.3 | 30.9 | 761040 |
| 4000 | 31.7 | 30.3 | 1895040 |

## 8.5 Results with the Deep Observation Function

Based on the fact that delta features may harm the generalization ability of HCNF, in this section, we did not use delta features for the deep observation function. We used 500 gates for each layer and all gates were shared by all states. Any regularizations were not used in these experiments. Table 7 shows the experimental results with the deep observation function. As the number of layers increases, we observe that the PERs decrease along with the Train-PERs. This result indicates that having a deep structure in the observation function increases the expression ability of the model while preserving the generalization ability. The best result in Table 7 was PER = 25.5 which was comparable to the best result of PER = 25.3 in Table 6 (b). By increasing the number of layers, the size of the parameters also increases. To confirm that the improvement was not caused by the increment of the parameter size only, we conducted an additional experiment where the number of gates was increased while the number of layers was fixed to one. The results are shown in Table 8. As can be seen in the table, increasing the number of gates, thereby increasing the number of parameters, did not improve the PERs significantly. From this result, we can conclude that the improvements of the PERs were not due to the increase of the parameter size, but rather due to adding more layers. Learning deep feed forward neural network with random initialization is considered a difficult problem [24]. Recently proposed sophisticated pre-training algorithms would be useful in providing reasonable initialization to HCNF [25]. Detailed explanation about deep architectures can be found in [26].

Finally, we compared HCNFs with the deep observa-

**Table 9** Comparison of the phoneme recognition results between Deep-HCNF and triphone-HMM on the TIMIT core test set [%].

| Model | PER | Train-PER | #Param |
|---|---|---|---|
| Deep-HCNF (w/o HB-MMI) | 25.0 | 16.2 | 2522040 |
| Deep-HCNF (w/ HB-MMI, $b = 3.0$) | 24.8 | 12.8 | |
| Triphone-HMM (MLE) | 27.3 | 21.9 | 504823 |
| Triphone-HMM (MPE) | 27.6 | 23.1 | |

**Table 10** Comparison of the phoneme recognition results between Deep-HCNF and triphone-HMM on the IPA100 test set [%].

| Model | PER | Train-PER | #Param |
|---|---|---|---|
| Deep-HCNF (w/o HB-MMI) | 13.1 | 7.8 | 4734965 |
| Deep-HCNF (w/ HB-MMI, $b = 1.0$) | 12.1 | 6.5 | |
| Triphone-HMM (MLE) | 14.1 | 8.9 | 5640178 |
| Triphone-HMM (MPE) | 12.4 | 6.4 | |

**Table 11** Computational complexities of HMM, HCRF and HCNF for a sample. F-B and DT stands for Forward-Backward and discriminative training, respectively. $T$, $|S|$, $M$, $D$, $K$ and $L$ are the number of frames of the sample, the number of states, the number of mixtures of a GMM, the number of features, the number of gates in (Deep-)HCNF and the number of layer in Deep-HCNF, respectively. In this table, we regarded that the number of gates of each layer in Deep-HCNF is all same and used $K$ instead of $K_l$ for all layers.

| Model | Score / Gradient | F-B / Viterbi |
|---|---|---|
| HMM-GMM (diagonal, DT) | $O(T \cdot |S| \cdot M \cdot D)$ | $O(T \cdot |S|^2)$ |
| HCRF | $O(T \cdot |S| \cdot D)$ | $O(T \cdot |S|^2)$ |
| HCNF | $O(T \cdot |S| \cdot K \cdot D)$ | $O(T \cdot |S|^2)$ |
| Deep-HCNF | $O(T \cdot L \cdot K^2)$ | $O(T \cdot |S|^2)$ |

tion function to the triphone-HMMs. In this experiment, we used a window length of 15 for both corpora. A layer size of 10 and a gate size of 500 were used for the experiment on the TIMIT corpus and a layer size of 5 and a gate size of 1024 were used for the experiment on the ASJ+JNAS corpus. Table 9 shows the result on the TIMIT corpus. Using the corpora, the Deep-HCNFs outperformed triphone-HMMs regardless of the use of HB-MMI. HB-MMI provided an additional improvement on the PER and yielded the result of PER = 24.8 which was the best result in this study on the TIMIT core test set. Table 10 shows the result on the ASJ+JNAS corpus. With HB-MMI, Deep-HCNF was superior to the result of the triphone-HMM trained in the MPE manner even though it employed the monophone structure. These results confirm the effectiveness of the deep observation function in HCNF.

### 8.6 Computational Complexity of Each Model

Table 11 summarizes the computational (time) complexities of HMM, HCRF, HCNF and Deep-HCNF for a sample. In a training stage, the score computation, gradient (or sufficient statistics for HMM) computation and Forward-Backward computation are required. On the other hand, in a decoding stage, the score computation and Viterbi computation are required. Forward-Backward and Viterbi algorithms require only $O(T \cdot |S|^2)$ for all models. They are negligible compared to the computational costs of the score and gradient computations. Therefore, we only discuss the score and gradient computations hereafter in this section.

Since the computational costs for score and gradient computations of HMM, HCRF and HCNF have all $T \cdot |S|$ in their equations, the differences come from only the remaining part, namely, $M \cdot D$, $D$ and $K \cdot D$ for HMM, HCRF and HCNF, respectively. Although the computational cost of HCRF looks much less than the computational cost of HMM, the feature dimension $D$ for HCRF usually becomes larger than the feature dimension $D$ for HMM. In our experiments, we used $D = 702$ for HCRF and $M \cdot D = 32 \cdot 39 = 1248$ for HMM. HCNF requires a $K$ times larger computational cost than HCRF due to the additional computations introduced by the gates. In our experiments, we used $K \cdot D = 4 \cdot 702 = 2808$ which is the biggest among three models. The actual training time for HCNF on the TIMIT corpus was 33 hours for 30 epochs while it was 7 hours for HCRF. We used a Linux machine which had a 2.53 GHz Intel Xeon CPU with 16 cores and a 48 GByte main memory to compute the computational times. Programs were executed using 8 threads. The MPE training of HMM took 7 hours using a single core (1 thread) for 10 iterations. The less computational time for the HMM was due to the approximation introduced by the use of lattices which restrict the hypotheses space. HCRF and HCNF did not use such lattices and computed everything without any approximations.

Deep-HCNF requires the highest computational cost $O(T(|S| \cdot K + L \cdot K^2 + K \cdot D))$. The second term $L \cdot K^2$ has the biggest impact and therefore Table 11 shows only the second term. To obtain the results shown in Table 9, the training of the Deep-HCNF took 126 hours even though we used NVIDIA Tesla c2070 to speed up the computation.

In our experiments, all models were able to run under real time in the test phase.

## 9. Conclusions

In this paper, we proposed HCNF for ASR that incorporated gate functions used in neural networks into HCRF and the three methods to further improve ASR by HCNF. The phoneme recognition results for HCNF on the TIMIT and ASJ+JNAS corpora outperformed the results for HCRF and the results of triphone HMMs trained in the MPE manner. HCNF could be trained successfully without either a reasonable initial model or initial alignment. By combining the HB-MMI and deep observation function, we achieved the best result of PER = 24.8 on the TIMIT core test set and PER = 12.1 on the IPA100 test set. We summarized the results obtained by the proposed HCNFs in Table 12 and 13. The result on the TIMIT core test which was much worse than the recently reported best results where PERs under 20% were achieved [27], [28]. However, as shown in Sect. 8.5, HCNF can use a deep feed forward neural network in the observation function, and therefore, we can use a sophisticated pre-training algorithm such as the deep belief network (DBN) to provide a deep observation function with reasonable initialization in HCNF, which might yield

**Table 12** Phoneme recognition results by proposed HCNFs on the TIMIT core test set [%].

| Model | PER | Train-PER | #Param |
|---|---|---|---|
| HCNF (w/ delta) | 28.0 | 16.1 | 412656 |
| HCNF (w/o delta) | 30.9 | 28.7 | 546480 |
| + HB-MMI | 28.0 | 16.1 | 546480 |
| + Hierarchical state posterior feature | 25.3 | 16.6 | 2936736 |
| Deep-HCNF (w/o delta) | 25.0 | 16.2 | 2522040 |
| + HB-MMI | 24.8 | 12.8 | 2522040 |

**Table 13** Phoneme recognition results by proposed HCNFs on the IPA100 test set [%].

| Model | PER | Train-PER | #Param |
|---|---|---|---|
| HCNF (w/ delta) | 15.7 | 14.3 | 412656 |
| + HB-MMI | 13.9 | 12.8 | 412656 |
| HCNF (w/o delta) | 19.1 | 19.1 | 546480 |
| + HB-MMI | 18.0 | 17.3 | 546480 |
| Deep-HCNF (w/o delta) | 13.1 | 7.8 | 4734965 |
| + HB-MMI | 12.1 | 6.5 | 4734965 |

PERs under 20% for HCNF on the TIMIT core test set. In other words, the proposed HCNF contains a DBN architecture [27].

In future studies, we need to examine HCNF using LVCSR experiments. A similar idea was examined in [14] where HCNF was used as a type of acoustic model. However, we will adapt HCNF to LVCSR directly. Furthermore, we need to deal with context dependencies in HCNF. The simplest way to incorporate context dependencies into HCNF is to use senones as stated in [29]. Exploring the effective features for HCNF is also of interest. In addition, as we have just described, we must incorporate a sophisticated pre-training method like DBN into HCNF to obtain better results than state-of-the-art results. Comparing HCNF with HCRF which uses mixtures is also required.

## Acknowledgements

Part of this research was supported by Global COE Program “Frontiers of Intelligent Sensing” from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

### References

[1] S. Furui, “Speaker-independent isolated word recognition using dynamic features of speech spectrum,” IEEE Trans. Acoust. Speech Signal Process., vol.ASSP-34, no.1, pp.52–59, Feb. 1986.

[2] S. Nakagawa and K. Yamamoto, “Speech recognition using hidden Markov models based on segmental statistics,” Systems and Computers in Japan, vol.28, no.7, pp.31–38, June 1997.

[3] N. Kanedera, T. Arai, H. Hermansky, and M. Pavel, “On the relative importance of various components of the modulation spectrum for automatic speech recognition,” Speech Commun., vol.28, pp.43–55, May 1999.

[4] D. Povey, Discriminative Training for Large Vocabulary Speech Recognition, Ph.D. thesis, Cambridge University Engineering Dept, 2003.

[5] H. Hermansky, D. Ellis, and S. Sharma, “Tandem connectionist feature stream extraction for conventional HMM systems,” Proc. ICASSP, vol.3, pp.1635–1638, 2000.

[6] J. Lafferty, A. McCallum, and F. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data,” Proc. 18th International Conference on Machine Learning, 2001.

[7] E. Fosler.-L. and J. Morris, “Crandem systems: Conditional random field acoustic models for hidden Markov models,” Proc. ICASSP, pp.4049–4052, 2008.

[8] A. Gunawardana, M. Mahajan, A. Acero, and J. Platt, “Hidden conditional random fields for phone classification,” Proc. Interspeech, pp.1117–1120, 2005.

[9] Y.H. Sung, C. Boulis, C. Manning, and D. Jurafsky, “Regularization, adaptation, and non-independent features improve hidden conditional random fields for phone classification,” Proc. ASRU, pp.347–352, 2007.

[10] Y.H. Sung and D. Jurafsky, “Hidden conditional random fields for phone recognition,” Proc. ASRU, pp.107–112, 2009.

[11] G. Heigold, D. Rybach, R. Schluter, and H. Ney, “Investigations on convex optimization using log-linear HMMs for digit string recognition,” Proc. ASRU, pp.216–219, 2009.

[12] J. Peng, L. Bo, and J. Xu, “Conditional neural fields,” Proc. Advances in Neural Information Processing Systems 22, pp.1419–1427, 2009.

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

[14] B. Kingsbury, “Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling,” Proc. ICASSP, pp.3761–3764, 2009.

[15] D. Povey and B. Kingsbury, “Evaluation of proposed modifications to MPE for large scale discriminative training,” Proc. ICASSP, pp.IV-321–IV-324, 2007.

[16] D. Povey, D. Kanevsky, and B. Kingsbury, “Boosted MMI for model and feature-space discriminative training,” Proc. ICASSP, pp.4058–4061, 2008.

[17] G. Heigold, T. Deselaers, R. Schlüter, and H. Ney, “Modified MMI/MPE: A direct evaluation of the margin in speech recognition,” Proc. ICML, pp.384–391, 2008.

[18] J. Pinto, S. Garimella, M.M.-Doss, H. Hermansky, and H. Bourland, “Analysis of MLP-based hierarchical phoneme posterior probability estimatior,” IEEE Trans. Audio Speech Language Process., vol.19, no.2, pp.225–241, 2011.

[19] A. Mohamed, G. Dahl, and G. Hinton, “Deep belief networks for phone recognition,” Proc. NIPS, pp.1–9, 2010.

[20] T.N. Sainath, B. Ramabhadran, and M. Picheny, “An exploration of large vocabulary tools for small vocabulary phonetic recognition,” Proc. ASRU, pp.359–364, 2009.

[21] M. Mahajan, A. Gunawardana, and A. Acero, “Training algorithms for hidden conditional random fields,” Proc. ICASSP, pp.I-273–I-276, May 2006.

[22] J. Duchi and Y. Singer, “Efficient learning using forward-backward splitting,” Proc. NIPS, pp.495–503, 2009.

[23] K.F. Lee and H.W. Hon, “Speaker-independent phone recognition using hidden Markov models,” IEEE Trans. Acoust. Speech Signal Process., vol.37, no.11, pp.1641–1648, Nov. 1989.

[24] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” Proc. AISTATS, pp.249–256, 2010.

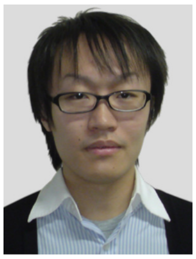
[25] D. Erhan, A. Courville, Y. Bengio, and P. Vincent, “Why does unsupervised pre-training help deep learning?,” Proc. AISTATS, pp.201–208, 2010.

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

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

[28] T.N. Sainath, D. Nahamoo, B. Ramabhadran, D. Kanevsky, V. Goel, and P.M. Shah, “Exemplar-based sparse representation phone identification features,” Proc. ICASSP, pp.4492–4495, 2011.

[29] G. Dahl, D. Yu, L. Deng, and A. Acero, "Large vocabulary continuous speech recognition with context-dependent DBN-HMMS," Proc. ICASSP, pp.4688–4691, 2011.



**Yasuhisa Fujii** graduated from Toyohashi University of Technology with his Bachelor's and Master's degrees in 2007 and 2009. Since 2009, he has been studying at Toyohashi University of Technology as a doctoral student. His research interest is in spoken language processing, signal processing, pattern recognition, and machine learning. He is a member of IPSJ, ASJ, and IEEE.

**Kazumasa Yamamoto** received his B.E., M.E. and Dr. Eng. degrees in Information and Computer Sciences from Toyohashi University of Technology, Toyohashi, Japan, in 1995, 1997 and 2000. From 2000 to 2007, he was a research associate in the Department of Electrical and Electronic Engineering, Faculty of Engineering, Shinshu University, Nagano, Japan. Since 2007, he has been an assistant professor in the Department of Information and Computer Sciences, Toyohashi University of Technology, Toyohashi, Japan. His current research interests include speech recognition and privacy protection for speech signals. He is a member of ASJ and IPSJ.

**Seiichi Nakagawa** received the Dr. of Eng. degree from Kyoto University in 1977. He joined the faculty of Kyoto University in 1976 as a Research Associate in the Department of Information Sciences. From 1980 to 1983, he was an Assistant Professor, from 1983 to 1990, he was an Associate Professor and since 1990, he has been a Professor in the Department of Information and Computer Sciences, Toyohashi University of Technology, Toyohashi. From 1985 to 1986, he was a Visiting Scientist in the Department of Computer Science, Carnegie-Mellon University, Pittsburgh, USA. He received the 1997/2001 Paper Award from the IEICE and the 1988 JC Bose Memorial Award from the Institution of Electro, Telecomm. Engrs. His major interests in research include automatic speech recognition/speech processing, natural language processing, human interface and artificial intelligence. He is a Fellow of the IPSJ.