International Journal on Document Analysis and Recognition (IJDAR)
https://doi.org/10.1007/s10032-018-0307-0

ORIGINAL PAPER

A comprehensive study of hybrid neural network hidden Markov model for offline handwritten Chinese text recognition

Zi-Rui Wang[1] · Jun Du[1] · Wen-Chao Wang[1] · Jian-Fang Zhai[2] · Jin-Shui Hu[2]

Received: 28 January 2018 / Revised: 29 May 2018 / Accepted: 2 June 2018

Abstract
This paper proposes an effective segmentation-free approach using a hybrid neural network hidden Markov model (NN-HMM) for offline handwritten Chinese text recognition (HCTR). In the general Bayesian framework, the handwritten Chinese text line is sequentially modeled by HMMs with each representing one character class, while the NN-based classifier is adopted to calculate the posterior probability of all HMM states. The key issues in feature extraction, character modeling, and language modeling are comprehensively investigated to show the effectiveness of NN-HMM framework for offline HCTR. First, a conventional deep neural network (DNN) architecture is studied with a well-designed feature extractor. As for the training procedure, the label refinement using forced alignment and the sequence training can yield significant gains on top of the frame-level cross-entropy criterion. Second, a deep convolutional neural network (DCNN) with automatically learned discriminative features demonstrates its superiority to DNN in the HMM framework. Moreover, to solve the challenging problem of distinguishing quite confusing classes due to the large vocabulary of Chinese characters, NN-based classifier should output 19900 HMM states as the classification units via a high-resolution modeling within each character. On the ICDAR 2013 competition task of CASIA-HWDB database, DNN-HMM yields a promising character error rate (CER) of 5.24% by making a good trade-off between the computational complexity and recognition accuracy. To the best of our knowledge, DCNN-HMM can achieve a best published CER of 3.53%.

Keywords Deep neural network · Deep convolutional neural network · Hidden Markov model · N-gram language model · Handwritten Chinese text recognition

1 Introduction

The robust recognition of handwritten text in an unconstraint writing style, especially for the Chinese language with a large vocabulary, is an important branch of artificial intelligence. Historically, many research efforts [1,2] were made on this challenging problem. In the recent competition of Chinese handwriting recognition [3,4], four tasks are explicitly defined, including online/offline handwritten Chinese character recognition (HCCR), and online/offline handwritten Chinese text recognition (HCTR). Obviously, the online recognition task is relatively easier than the offline recognition task as the ink trajectory information can be leveraged. Meanwhile, HCTR is more challenging than HCCR due to the free-style writing. In this study, we focus on the most difficult task, namely the offline HCTR.

One key issue for offline HCTR is the modeling of the text line, which seems more challenging than the classifier design of isolated characters by the comparison of recognition performance between offline HCCR and HCTR tasks in the ICDAR competition. Accordingly, the existing techniques can be divided into two categories: oversegmentation-based and segmentation-free approaches. The former approaches [5–8] typically performed an oversegmentation of the text

✉ Jun Du
jundu@ustc.edu.cn

Zi-Rui Wang
cs211@mail.ustc.edu.cn

Wen-Chao Wang
wangwenc@mail.ustc.edu.cn

Jian-Fang Zhai
jfzhai@iflytek.com

Jin-Shui Hu
jshu@iflytek.com

[1] University of Science and Technology of China, Hefei, Anhui, People's Republic of China

[2] iFlytek Research, Hefei, Anhui, People's Republic of China

Published online: 15 June 2018

line followed by the decoding with a character classifier and the information of linguistic/geometric contexts. In [8], multiple contexts were integrated to achieve the best performance on the ICDAR 2013 competition task. More recently, Wang et al. [9] used positive and negative samples to train the so-called heterogeneous CNN as the character classifier. Wu et al. [10] adopted three different CNN models to replace conventional character classifier, oversegmentation, and geometric models, respectively. By contrast, not many segmentation-free approaches were proposed. One representative approach in [11] used the Gaussian mixture model-based hidden Markov model (GMM-HMM) as a recognizer. Recently, the artificial intelligence and image analysis (Ai2A) laboratory has adopted the multidimensional long short-term memory recurrent neural network (MDLSTM-RNN) for offline HCTR [12]. The MDLSTM-RNN is extended from LSTM-RNN [13], which was successfully applied for the handwriting recognition tasks of Western languages with a small set of character classes. This end-to-end approach is quite flexible as the explicit segmentation is avoided by using connectionist temporal classification (CTC) technique [14]. Another recent work [15] utilized a CNN followed by an LSTM neural network under the HMM framework to obtain a significant improvement when compared with LSTM-HMM model. Generally speaking, the main advantage of the segmentation-free method is the explicit segmentation part that is not necessary, which can potentially avoid both the propagation of the segmentation errors to the subsequent character classifier and the rule settings with a bunch of tuning parameters commonly used in the segmentation part. And the segmentation rule setting in the oversegmentation-based approach can vary for different languages according to the geometric structure of characters.

In this study, an effect segmentation-free framework via a hybrid neural network hidden Markov model (NN-HMM) is proposed for offline HCTR. In the general Bayesian framework, the text line is sequentially modeled by HMMs with each representing one character class. Meanwhile, the NN-based classifier is adopted to calculate the posterior probability of all HMM states. The key issues in feature extraction, character modeling, and language modeling are comprehensively investigated to show the effectiveness of NN-HMM framework for offline HCTR. In comparison with the GMM-HMM as a generative model in [11,16], the proposed DNN-HMM and DCNN-HMM as discriminative models can better handle the dimension correlation of input feature vector and have a powerful modeling capability with deeper structures. Furthermore, the main difference of our approach from the MDLSTM-RNN [12] is that HMM is used to model the text line rather than the CTC technique. Our experiments on the ICDAR 2013 competition task demonstrate the superiority of the proposed approach to both the best previously reported oversegmentation-based and segmentation-free approaches [10,12]. In [15,17–19], NN-HMMs are also adopted in different research areas. We make a comparison of our approach with two representative and related work in [15,17]. The main differences from our work can be summarized as follows. First, [17] focuses on a small vocabulary with only 11 digital classes. By contrast, the problem in this study is much more challenging for distinguishing thousands of Chinese character classes, which is obviously non-trivial. Second, for both NN-HMMs used in [15,17], the CNN or CNN+LSTM is adopted to estimate the label of the character classes rather than HMM states, which are actually similar to CTC. However, the high-resolution modeling using HMM states is one main contribution of this study, e.g., reducing the CERs from 12.33 (1-state HMM) to 7.16% (5-state HMM) in the DNN-HMM approach. Accordingly, the CNN architectures and the training and decoding procedures of NN-HMM are also quite different. Finally, the experimental designs of our work, including examining the key issues of feature extraction, character modeling, and language modeling, are much more compressive. All these factors lead to the large performance gap (CER: 3.53 vs. 16.5%) between our best approach and the best CNN-LSTM-HMM model in [15] on the same competition set of CASIA-HWDB database. Even our well-designed DNN-HMM without using N-gram ($N = 0$) language model (LM) outperforms the best model in [15].

This work is an extension of the recently conference papers [20,21] with the main contributions: (1) more technical details of feature extraction, character modeling, and language modeling are elaborated, (2) the proposed sequence training for DNN-HMM generates a remarkable gain over the frame-level cross-entropy (CE) criterion, (3) the well-designed DCNN-HMM is adopted to further improve the recognition accuracy, and (4) more experiments are designed with the analyses to emphasize the contributions of the important components to the overall performance.

The remainder of the paper is organized as follows. In Sect. 2, we first give an overview of the proposed framework. In Sect. 3, the feature extraction is introduced. In Sects. 4 and 5, we describe the character and language modeling. In Sect. 6, we report experimental results and analyses. Finally, we make the conclusion in Sect. 7.

2 The overall framework

The main principle of the proposed new framework is based on the Bayesian formula:

$$\begin{aligned}\hat{\mathbf{C}} &= \arg\max_{\mathbf{C}} p(\mathbf{C} \mid \mathbf{X}) \\ &= \arg\max_{\mathbf{C}} p(\mathbf{X} \mid \mathbf{C}) P(\mathbf{C}) \end{aligned} \tag{1}$$

where $\mathbf{X} = \{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$ is a $(T+1)$-frame feature sequence of one handwritten text line and $\mathbf{C} = \{C_1, C_2, \ldots, C_n\}$ is the underlying n-character sequence. $p(\mathbf{X} \mid \mathbf{C})$, which can be named as the character model, is the conditional probability of $\mathbf{X}$ given $\mathbf{C}$ corresponding to a sequence of HMMs. Each HMM with a set of states represents one character class. Meanwhile, $P(\mathbf{C})$, namely the language model, is the probability of n-character sequence $\mathbf{C}$. The value of n varies dynamically in the decoding process by combining the scores of the character model and language model. And $\mathbf{C}$ is just one sequence from a hypotheses space of all candidates.

Specifically, the implementation of the Bayesian framework in this study could be illustrated in Fig. 1. In the training stage, three HMM systems, including GMM-HMM, DNN-HMM, and DCNN-HMM, are constructed. First, for GMM-HMM and DNN-HMM, the gradient-based features on a frame basis are extracted from the image samples, which is followed by principal component analysis (PCA) transformation to obtain a lower- dimensional feature vector. Meanwhile, for DCNN-HMM, only framing with a sliding window is performed on the raw images before training. Then, maximum likelihood estimation (MLE) is used for the parameter learning of GMM-HMMs for all character classes. With the well-trained GMM-HMMs, we obtain the frame-level labels via the state-level forced alignment for the subsequent DNN/DCNN training. As for the DNN-HMM system, we train the DNN model using the CE criterion [22].

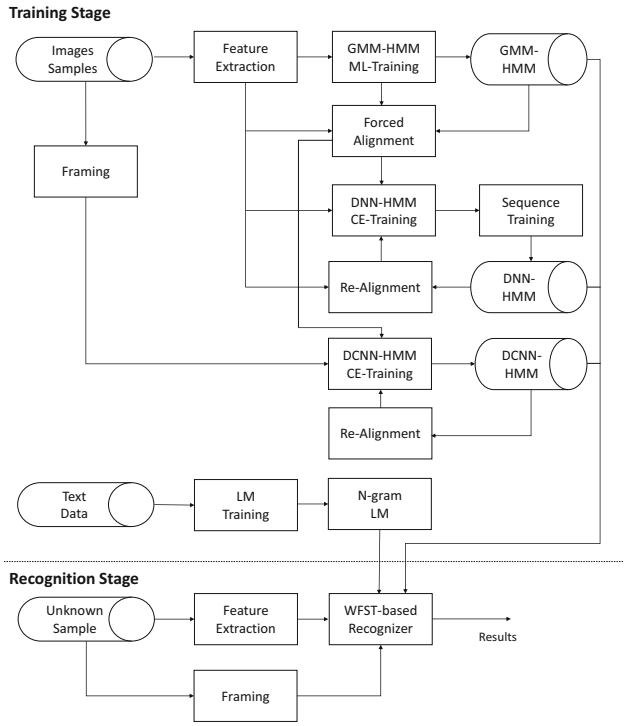

Fig. 1 Overall development flow and architecture

To refine the labels initialized by GMM-HMM system, the realignment in a manner of multi-pass training can be adopted for the DNN-HMM system. To further improve the recognition accuracy, the sequence training [23] is conducted on top the CE training. For the DCNN-HMM system, we first train our DCNN model with the state-level labels obtained from the well-trained GMM-HMM system and an enhanced language model. And then, the batch normalization [24] and realignment are used to help further improve the recognition accuracy. The same CE criterion is used for training DCNN-HMM. On the other hand, the widely used N-gram LM [25] is built using the collected text data.

In the recognition stage, the final recognition results can be generated via a weighted finite-state transducer (WFST)-based [26,27] decoder by integrating both GMM-HMM/DNN-HMM/DCNN-HMM and N-gram LM. In the following sections, we elaborate the details of feature extraction, character modeling, and language modeling.

3 Feature extraction

The procedure of feature extraction for the GMM-HMM and DNN-HMM systems consists of the following steps.

Step 1 Text line preprocessing
First, the Otsu's method is used for binarization of the original text line image. Then, the height of text line will be estimated, followed by the size normalization to 60 pixels while keeping the aspect ratio. And the margin (40 pixels for left and right, 100 pixels for top and bottom) is extended to accommodate the text area for all the sliding windows in the next step. Accordingly, the centerline is calculated.

Step 2 Framing
Along the centerline, each frame, represented by a $80 \times F_l$ sliding window (normalized to 64×32 for the following steps) from the left to right, with a frame length of F_l pixels and frame shift of F_s pixels, is scanned across the text line.

Step 3 Calculating 8-directional gradient images
In each frame, a Hanning filter is first applied to convert the binary image to the gray image, and then, the gradient features via the Sobel operator are calculated and 8-directional gradient images can be generated according to [28,29].

Step 4 Spatial sampling and blurring
Each gradient image is divided uniformly into 8×4 grids with the corresponding centers treated as locations of 8×4 spatial sampling points. Then, the blurring via the Gaussian filters is applied as in [30]. The spatial sampling operation is to reduce the data dimension of each directional gradient image from 64×32 to 8×4, while the Gaussian blurring aims to generate more effective and noise-insensitive features

according to [30]. Based on all 8 gradient images, a 256-dimensional feature vector is formed.

Step 5 PCA
In the last step, PCA transformation [31] is adopted to obtain a lower D-dimensional feature vector fed to the character modeling.

As for the DCNN-HMM system, only the first two steps of the above procedure are applied for framing.

4 HMM-based character model

The offline handwritten Chinese text line is a sequence of Chinese characters. In this study, we adopt a left-to-right HMM to model one character class as a basic unit. Accordingly, the text line is modeled by a sequence of HMMs. As shown in Fig. 2, each character is represented by a left-to-right HMM [32,33] with several hidden states; then, a text line can be modeled by cascaded character HMMs. For the feature sequence extracted from one handwritten character sample, each frame is supposed to be assigned to one underlying state. For each state, an output distribution describes the statistical property of the observed feature vector. With HMMs, the $p(\mathbf{X} \mid \mathbf{C})$ in Eq. (1) can be decomposed as:

$$\begin{aligned} p(\mathbf{X} \mid \mathbf{C}) &= \Sigma_S \left[p(\mathbf{X} \mid S, \mathbf{C}) p(S \mid \mathbf{C}) \right] \\ &= \Sigma_S \left[\pi(s_0) \prod_{t=1}^{T} a_{s_{t-1} s_t} \prod_{t=0}^{T} p(\mathbf{x}_t | s_t) \right] \end{aligned} \tag{2}$$

where $\pi(s_0)$ is the prior probability of the initial state s_0 and $a_{s_{t-1}s_t}$ is the transition probability from state s_{t-1} at the $(t-1)$th frame to state s_t at the tth frame. $S = \{s_0, s_1, \ldots, s_T\}$ is one underlying state sequence of $\mathbf{C}$ to represent $\mathbf{X}$. The number of hidden states for each HMM is N_S. The key item in Eq. (2) is $p(\mathbf{x}_t|s_t)$ denoting the observation probability. If a generative model is employed, e.g., GMM-HMM [11], then $p(\mathbf{x}_t|s_t)$ can be directly calculated by a GMM. Otherwise, it can be rewritten as:

$$p(\mathbf{x}_t|s_t) = \frac{p(s_t|\mathbf{x}_t) p(\mathbf{x}_t)}{p(s_t)} \tag{3}$$

where $p(s_t|\mathbf{x}_t)$ is the state posterior probability, $p(s_t)$ is the prior probability of each state estimated from the training set, and $p(\mathbf{x}_t)$ is independent of the character sequence which can be ignored in the optimization of Eq. (1). In this case, a discriminant function, e.g., DNN or CNN, is adopted to compute $p(s_t|\mathbf{x}_t)$, which can be used to estimate $p(\mathbf{x}_t|s_t)$ via Eq. (3). In the following subsections, three HMMs are introduced.

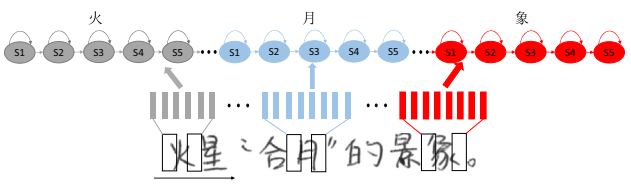

Fig. 2 Illustration of text line modeled by cascaded character HMMs

4.1 GMM-HMM

For GMM-HMM system, $p(\mathbf{x}_t|s_t)$ is represented by a GMM:

$$\begin{aligned} p(\mathbf{x}_t|s_t) = \sum_{m=1}^{M_{s_t}} \Bigg\{ & \frac{\omega_{s_t m}}{\sqrt{(2\pi)^D |\mathbf{\Sigma}_{s_t m}|}} \\ & \times \exp\left[-\frac{1}{2} (\mathbf{x}_t - \boldsymbol{\mu}_{s_t m})^\top \mathbf{\Sigma}_{s_t m}^{-1} (\mathbf{x}_t - \boldsymbol{\mu}_{s_t m}) \right] \Bigg\} \end{aligned} \tag{4}$$

where $\omega_{s_t m}$, $\boldsymbol{\mu}_{s_t m}$, and $\mathbf{\Sigma}_{s_t m}$ are the weight, mean vector, and covariance matrix of m-th component for the output distribution of s_t with M_{s_t} Gaussian mixture components. $\mathbf{\Sigma}_{s_t m}$ is typically a diagonal covariance. The parameters of GMM-HMMs for all character classes are learned according to the maximum likelihood estimation [34]. By adopting Baum–Welch or Viterbi training algorithm [35], the state prior probabilities and transition probabilities of HMMs, and the weight/mean/covariance parameters of GMMs, can be effectively estimated in an iterative manner and the Gaussian mixtures of all character classes can be progressively learned and increased from scratch [35,36].

4.2 DNN-HMM

DNN-HMM can achieve much more powerful modeling capability than GMM-HMM due to several reasons. First, GMM is a generative model, while DNN is a discriminative model which can better match the evaluation metrics of the recognition task. Second, DNN can fully utilize the high-dimensional multiple-frame input (the center frame plus the left/right neighboring frames) to make a better prediction, while GMM often uses one-frame input as it cannot model high-dimensional data well especially with the diagonal covariance. Third, GMM typically makes a strong assumption of diagonal covariances because the singularity problem exists for the full covariances especially with high-dimensional data. However, DNN can well handle this using the fully connected layers.

For the GMM-HMM training, the frame-level labels are not necessary as an embedded re-estimation procedure could be applied with the character-level labels [35]. However, for the DNN-HMM system, to calculate the state posterior $p(s_t|\mathbf{x}_t)$ in Eq. (3), the frame-level state labels are generated via a general-purpose Viterbi recognizer with GMM-HMMs [35] for the subsequent training of DNN model. To conduct a fair comparison, the HMM topology of DNN-HMM system

is directly copied from that of GMM-HMM system, including the corresponding state prior probabilities and transition probabilities in Eq. (2). Accordingly, we design the following procedure for the parameter learning via the CE criterion and the sequence training:

Step 1 DNN CE training
As shown in Fig. 3, the DNN adopted for character modeling is a feed-forward, artificial neural network with many layers of hidden units between its input and output which aims to model the posterior probability of the HMM states using the input feature vector of multiple neighboring frames. The hidden units between two consecutive layers are fully connected. After the random initialization of all the weight parameters, a supervised fine-tuning is conducted by minimizing the following cross-entropy function:

$$\mathcal{F}_{\mathrm{CE}}(\mathbf{W}, \mathbf{b}) = -\sum_{l=1}^{L}\sum_{t=1}^{T_l} \log p(s_t^l|\mathbf{x}_{t\pm\tau}^l, \mathbf{W}, \mathbf{b}) \tag{5}$$

where $p(s_t^l|\mathbf{x}_{t\pm\tau}^l, \mathbf{W}, \mathbf{b})$ is the estimated posterior probability of the target state s_t^l from the DNN output given the input feature vector $\mathbf{x}_{t\pm\tau}^l$ with the expansion of $2\tau+1$ neighboring frames. L is the number of handwritten text lines, while T_l is the frame number of the lth line. $\mathbf{W}$ and $\mathbf{b}$ denote all the parameters of weight matrices and bias vectors, respectively. The standard stochastic gradient descent (SGD) algorithm is used to optimize the objective function.

Step 2 DNN sequence training
The CE training does not explore the relationship among the frames of output layer and the long-term discrimination between character classes across the text line is not considered. To address this issue, the sequence training is proposed by maximizing a general objective function:

$$\mathcal{F}_{\mathrm{ST}}(\mathbf{W}, \mathbf{b}) = \sum_{l=1}^{L} f\left(\frac{\sum_{\mathbf{C}} p^{\kappa}(\mathbf{X}_l|\mathbf{C})P(\mathbf{C})A(\mathbf{C}, \mathbf{C}_l)}{\sum_{\mathbf{C}} p^{\kappa}(\mathbf{X}_l|\mathbf{C})P(\mathbf{C})}\right) \tag{6}$$

where $\mathbf{X}_l$ and $\mathbf{C}_l$ are the sequences of feature vectors and character labels for the lth line, respectively; $\mathbf{C}$ is a hypothesis sequence corresponding to $\mathbf{X}_l$ from the lattice [23]; κ is a scaling factor for the character model $p(\mathbf{X}_l|\mathbf{C})$ to make a balance with language model $P(\mathbf{C})$; f is a smoothing function, while $A(\mathbf{C}, \mathbf{C}_l)$ is a gain function. With different settings of f and $A(\mathbf{C}, \mathbf{C}_l)$, several criteria are investigated in this study, namely maximum mutual information (MMI) [37], minimum word error (MWE) [38], and state-level minimum Bayes risk (sMBR) [39] as illustrated in Table 1. MMI aims to optimize the parameters from the perspective of information theory, while MWE and sMBR minimize the average error of the lines calculated in terms of different temporal resolutions.

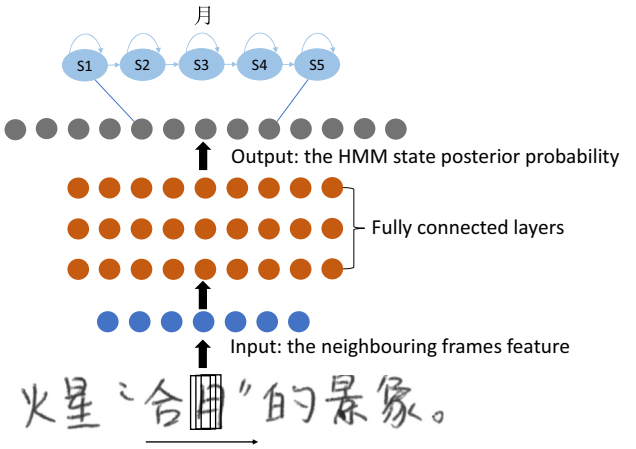

Fig. 3 The DNN architecture

Table 1 The criteria for sequence training

Criterion	f	$A(\mathbf{C}, \mathbf{C}_l)$	Objective
MMI	log	$\delta(\mathbf{C}, \mathbf{C}_l)$	Mutual information
MWE	1	$\lvert\mathbf{C}_l\rvert$-LEV$(\mathbf{C}, \mathbf{C}_l)$	Character accuracy
sMBR	1	$\lvert\mathbf{S}_l\rvert$-LEV$(\mathbf{S}, \mathbf{S}_l)$	State accuracy

$\delta(\cdot)$ is the Kronecker delta function. $|\cdot|$ denotes the number of symbols in a string. LEV(,) is the Levenshtein distance between two symbol strings

Step 3 Realignment
Due to the limited modeling capability of GMM-HMM, the labels generated from GMM-HMM are not always accurate. To refine the state-level labels, the DNN trained in **Step1** or **Step2** can be used for the realignment of all training data samples.

Step 4 Go to **Step1** or **Step2** several times.

4.3 DCNN-HMM

The main difference of DCNN-HMM from DNN-HMM is that $p(s_t|\mathbf{x}_t)$ in Eq. (3) is calculated from a deep CNN rather than a DNN. DNN uses a handcrafted feature extractor to generate a relatively low-dimensional vector from the high-dimensional raw image with the risk of information loss. However, CNN adopts the convolution layers to directly extract more discriminative features with the guidance of training criterion for the recognition tasks. As shown in Fig. 4, the DCNN successively consists of stacked convolutional layers optionally followed by spatial pooling, one or more fully connected layers and a softmax layer. For the convolutional and the pooling layers, each layer of them is a three-dimensional tensor organized by a set of planes called feature maps, while the fully connected layer and the softmax layer are the same as the conventional DNN. Inspired by the locally sensitive, orientation-selective neurons in the visual

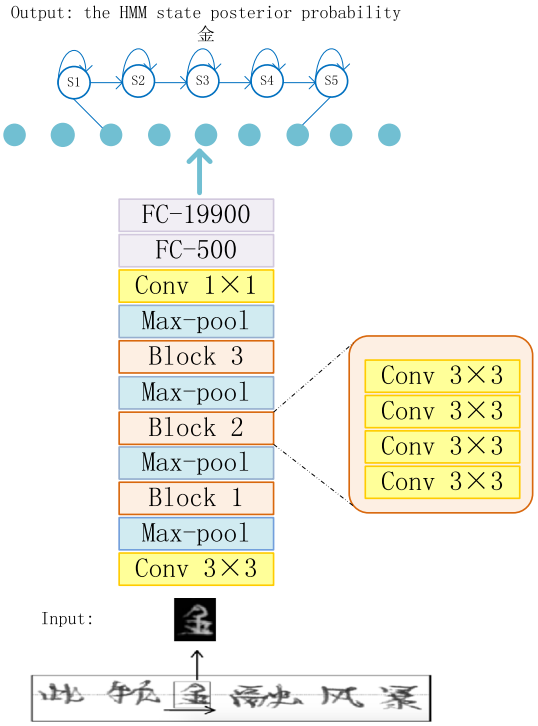

Fig. 4 The DCNN-HMM architecture

system of cats [40], each unit in a feature map is constrained to connect a local region in the previous layer, which is called the local receptive field. All units in the same feature map of a convolutional layer share a set of weights, each computing a dot product between its weights and local receptive field in the previous layer and then followed by nonlinear activation functions (ReLU). Meanwhile, the units in a pooling layer perform a spatial average or max operation for their local receptive field to reduce the spatial resolution and the noise interferences.

To further improve the modeling capability, bach normalization (BN) [24] is also investigated. To handle the change of the distribution in each layer, the BN operation simply normalizes each input node x_i of a layer and then two corresponding learnable parameters γ_i and β_i are used to scale and shift the normalized value, respectively:

$$\tilde{x}_i = \frac{x_i - \mu_i}{\sqrt{\sigma_i^2 + \varepsilon}} \tag{7}$$

$$y_i = \gamma_i \tilde{x}_i + \beta_i \tag{8}$$

where μ_i is the mean and σ_i^2 is the variance for the input node i. ε is a constant for numerical stability. In the training stage, μ_i and σ_i are estimated from every mini-batch, while they are replaced by the population statistics and unchanged in the test stage. As suggested in [24], we equip the BN operation for the outputs before nonlinearity in every convolutional layer and the nodes in the same feature map share the same parameters $\mu, \sigma^2, \gamma, \beta$.

The main contributions of GMM-HMM and DNN-HMM can be described as follows. First, the initial state-level labels and the HMM transition matrices should be obtained by GMM-HMM for NN-HMM. Second, based on the experiments of GMM-HMM and DNN-HMM, we can choose appropriate hyperparameters, e.g., frame length, frame shift, state number, and training strategy for DCNN-HMM. Finally, we can observe the discriminative models with deep structures can significantly improve performance over the generative model (GMM-HMM).

5 N-gram-based language model

The probability of the character sequence $\mathbf{C} = \{C_1, C_2, \ldots, C_n\}$, namely the language model $P(\mathbf{C})$ in Eq. (1), could be expressed as:

$$P(\mathbf{C}) = \prod_{i=1}^{n} P(C_i | C_{i-1}, C_{i-2}, \ldots, C_1). \tag{9}$$

However, the term $P(C_i|C_{i-1}, C_{i-2}, \ldots, C_1)$ cannot realistically depend on all $i-1$ conditioning histories $C_1, C_2, \ldots, C_{i-1}$ as the number of these values V^i for even a moderate vocabulary size V is too large to be accurately estimated. Accordingly, the so-called N-gram LM is proposed. If $N = 1$, $P(C_i|C_{i-1}, C_{i-2}, \ldots, C_1) = P(C_i)$, namely 1-gram (or *unigram*). For $N = 2$ and $N = 3$, we have the corresponding 2-gram (*bigram*) and 3-gram (*trigram*). Obviously, a higher-order N leads to a more powerful language model which can significantly improve the recognition accuracy. One key problem for estimating high-order N-gram LM is the data sparseness of training corpus. To address this problem, the N-gram smoothing [41] is often applied. In this work, we adopt the Katz smoothing [42]. The SRILM toolkit [43] is employed to generate Katz N-gram LM with different orders. Without using N-gram ($N = 0$), the recognition accuracy might be sharply declined as shown in the experiments.

Finally, by integrating both character and language models, the WFST-based decoder [26,27] is implemented to generate the final recognition results using the Kaldi toolkit [36].

6 Experiments

We conducted the experiments on a widely used database for HCTR released by the Institute of Automation of Chinese Academy of Sciences (CASIA) [44,45]. To train the character models, both offline isolated handwritten Chinese

character datasets (HWDB1.0 and HWDB1.1) and offline handwritten Chinese text datasets (HWDB2.0, HWDB2.1, and HWDB2.2) were used.

The ICDAR 2013 competition set was adopted as the evaluation set [4]. The character samples out of vocabulary were always recognized as the errors which were included in calculating the CER.

In the first set of experiments from Sects. 6.1 to 6.3, the conventional DNN is used to calculate the posterior probability of all HMM states for each frame-level feature vector which is extracted from a left-to-right sliding window along with the text line. Three key components, including feature extraction, character modeling and DNN training strategy, are comprehensively explored.

For the handcrafted feature extraction, the impacts of final feature dimension D, frame length F_l, frame shift F_s, and frame expansion τ would be discussed in Sect. 6.1. The frame expansion is not used for GMM-HMM system as GMM could not well model the high-dimensional data due to the diagonal covariance assumption.

For character modeling, we employed a left-to-right HMM with N_S states to represent each character class. In GMM-HMM system, a GMM with 40 Gaussian mixtures was denoted as the output distribution of each state. For both DNN and CNN, the size of output layer was $3.980 \times N_S$ corresponding to the number of states for all character classes. 3.980 is the total number of classes including Chinese characters, symbols, garbage.

For DNN training, the mini-batch size was 256. The initial learning rate of CE training was set to 0.008 and halved if the loss of cross-validation set was reduced during 16 iterations. For all criteria of sequence training, the learning rate was set to 0.000001 and the scaling factor κ in Eq. (6) was 0.9090 (default value in Kaldi toolkit). The 1-gram was used to perform the decoding to generate the hypotheses (lattices) in Eq. (6). Besides, we used the best DNN structure in [20] for the following experiments, i.e., 2048 hidden units were used for each of 6 hidden layers.

In order to be consistent with [20] , the same corpora were used to train our N-gram LM in the first set of experiments, including 93 MB texts of Sina News, 115 MB texts of People's Daily between 2000 and 2004, 208 MB texts of Guangming Daily between 1994 and 1998, 129 MB texts of other newspapers and the transcriptions of CASIA database.

To further improve the recognition rate in the second set of experiments in Sect. 6.4, DCNN was used to replace the conventional DNN with an enhanced 4-gram language model trained with more text data (including the above transcriptions and 63MB texts of 2011 People's Daily), and then, the effectiveness of BN transform and the realignment was verified. We used the Caffe toolkit [46] to implement the DCNN. The mini-batch size was 400, while the momentum was 0.9 and the weight decay was 0.0001. The learning rate was initially set to 0.01 and then decreased by 0.92 after every 10,000 iterations. 1.8 million iterations were conducted.

Table 2 The CER (in %) comparison of different dimensions after PCA transformation in DNN-HMM systems for different LMs on the evaluation set

	$D = 30$	$D = 50$	$D = 150$
0-g	18.24	**17.03**	18.13
2-g	10.41	**9.16**	9.47
3-g	8.28	**7.16**	7.38

In all the above experiments, the character error rate was adopted as the evaluation measure, which was defined as the ratio between the total number of substitution/insertion/deletion errors and the total number of character samples in the evaluation set:

$$\mathrm{CER} = \frac{N_\mathrm{s} + N_\mathrm{i} + N_\mathrm{d}}{N_t} \tag{10}$$

where N_t is the total number of character samples in the evaluation set. N_s, N_i, and N_d denote the numbers of substitution errors, insertion errors and deletion errors, respectively.

6.1 Experiments on feature extraction

The first set of experiments were designed to examine the impacts of settings in feature extraction to the recognition performance of DNN-HMM systems, including the dimension of final feature vector, frame expansion, length, and shift.

6.1.1 PCA

Table 2 lists a performance comparison of different dimensions after PCA transformation in DNN-HMM systems on the evaluation set. The bold in Table represents the best performance under the corresponding comparable conditions, which is the same meaning in the following Tables. In this experiment, frame length F_l was 40, frame shift F_s was 3, and frame expansion τ was 5. The state number N_S was 5; 2048 hidden units were used for each of 6 hidden layers. If D was too small, then it might lead to the information lost. Otherwise, the DNN might not handle the high-dimensional input vector well. So $D = 50$ setting (the default value thereafter) achieved a good trade-off across all LMs, namely 0-gram (actually decoding without LM), 2-gram, and 3-gram. From Table 2, the use of high-order LM was crucial to the recognition accuracy. A relative CER reduction of 58.0% was achieved by the system using 3-gram over the system with no LM (0-gram) under the setting of $D = 50$.

Table 3 The CER (in %) comparison of different frame expansions for different LMs on the evaluation set

	$\tau = 0$	$\tau = 1$	$\tau = 3$	$\tau = 5$	$\tau = 7$
0-g	20.21	17.52	**17.01**	17.03	17.92
2-g	8.99	**8.17**	8.59	9.16	9.80
3-g	7.01	**6.38**	6.73	7.16	7.62

Table 4 The CER (in %) comparison of different frame lengths for different LMs on the evaluation set

	$F_l = 20$	$F_l = 40$	$F_l = 80$
0-g	18.36	**17.01**	20.57
2-g	8.84	**8.59**	13.29
3-g	7.03	**6.73**	10.55

Table 5 The CER (in %) comparison of different frame shifts for different LMs on the evaluation set

$F_s = 2$			$F_s = 3$		
0-g	2-g	3-g	0-g	2-g	3-g
17.65	**8.51**	**6.61**	**17.01**	8.59	6.73

6.1.2 Frame expansion, length, and shift

One advantage of DNN-HMM in comparison with GMM-HMM is that it can learn the long context information via the frame expansion in the input layer to better estimate the underlying state posterior. In Table 3, with the same configuration as Table 2, we could observe that both $\tau = 1$ and $\tau = 3$ yielded better recognition results than the case without frame expansion. Nevertheless, larger τ (> 3) degraded the performance due to the problem of curse of dimensionality. It was interesting that $\tau = 3$ outperformed $\tau = 1$ without LM which was reasonable as the frame expansion played a similar role of LM by the awareness of more context information. As the character modeling is the focus of this study, the $\tau = 3$ is set as default for the following experiments. The frame length was set to 40 based on the comparison in Table 4. The frame shift was set to 3 based on the comparison in Table 5, which not only outperformed $F_s = 2$ for 0-gram case but also generated a smaller amount of frames to improve the training and decoding efficiency.

6.2 Experiments on character modeling

We conducted the second set of experiments to discuss the issues in character modeling, namely the impact of the number of HMM states.

The difference between HMM and CTC is that HMM can model the temporal information of a character via the so-called hidden states, while CTC usually combines RNN to utilizing temporal information. The one-state HMM is similar to CTC [12] aiming at modeling the text line in the character level. Table 6 shows the comparison between GMM-HMM and DNN-HMM systems with different settings of hidden states. Please note that this experiment was based on the best setting in Table 2. First, DNN-HMM systems consistently and significantly outperformed GMM-HMM systems for all state settings and LMs, which demonstrated the powerful modeling capability of DNN. In the best configuration ($N_S = 5$ and 3-gram), a relative CER reduction of 64.3% could be achieved by DNN-HMM over GMM-HMM. Second, HMM systems using only one state generated much worse performance than systems using multiple states, which implied that it was necessary to use different distributions for modeling the inner variations within one character class. Third, DNN-HMM with $N_S = 5$ obtained the lowest CERs for all LMs. Using more states ($N_S > 5$) could not further improve the recognition accuracy, which might be explained in two aspects. One was that the sample of one character usually consisted of limited frames corresponding to limited states, while the other one was overfitting problem with limited training data.

6.3 Experiments on DNN training strategy

In this section, we discussed the issues in DNN training strategy, including , label refinement, and sequence training.

6.3.1 Label refinement

The state-level labels play an important role in DNN training. In Table 7, we compare the DNN-HMM systems with multiple-pass realignments on the evaluation set. Obviously, the DNN-HMM trained with GMM-HMM initialized labels (No-ReFA) performed the worst. For different LMs, multiple-pass realignments could always reduce the CER. One interesting observation was that the weak LM required more passes of realignment than the strong LM to achieve the best performance. For example, three passes of realignment (ReFA-3) were needed for 0-gram system, while only one-pass realignment (ReFA-1) was necessary for 3-gram system. This was reasonable as the realignment of the state-level labels was equivalent to improving the accuracy of the context information, which was similar to the use of a strong LM. Overall, the realignment could bring a relative CER reduction of about 5%.

6.3.2 Sequence training

Based on the ReFA-3 setting in Table 7, the sequence training was conducted via different criteria, as shown in Table 8. After the first-pass sequence training, the realignment based on the new DNN-HMM was performed, which was followed

Table 6 The CER (in %) comparison of different hidden states between GMM-HMM and DNN-HMM systems for different LMs on the evaluation set

# of states	$N_S = 1$		$N_S = 4$		$N_S = 5$		$N_S = 6$		$N_S = 8$	
HMM model	GMM	DNN	GMM	DNN	GMM	DNN	GMM	DNN	GMM	DNN
0-g	51.64	23.07	37.25	17.56	35.24	**17.03**	34.98	17.10	34.89	17.80
2-g	37.60	14.14	23.87	9.48	23.46	**9.16**	23.39	9.25	24.86	10.33
3-g	34.76	12.33	20.38	7.38	20.04	**7.16**	20.20	7.37	21.98	8.42

Table 7 The CER (in %) comparison of multiple-pass realignments of DNN-HMM system for different LMs on the evaluation set

	No-ReFA	ReFA-1	ReFA-2	ReFA-3	ReFA-4
0-g	17.01	16.31	16.20	**16.11**	16.25
2-g	8.59	8.15	**7.80**	7.98	8.03
3-g	6.73	**6.37**	6.45	6.50	6.48

Table 8 The CER (in %) comparison of different criteria of DNN-HMM sequence training for different LMs on the evaluation set

Criterion	First-pass			Second-pass		
	MMI	MWE	sMBR	MMI	MWE	sMBR
0-g	15.74	15.57	15.55	15.64	**15.20**	15.55
2-g	7.66	7.85	7.75	7.57	**7.57**	7.85
3-g	6.07	6.22	6.22	6.01	**5.93**	6.22

by the second-pass sequence training. It was clear that the sequence training consistently improved the performance of the CE training with realignments. Compared with the ReFA-3 results in Table 7, relative CER reductions of 5.6, 5.1, and 8.8% were obtained for 0-gram, 2-gram, and 3-gram, respectively.

6.4 Experiments on DCNN-HMM system

In this section, we focused on two parts. First, we used DCNN model to computer the posterior probability of the HMM states for each frame image. As shown in Fig. 4, there were 16 weight layers (14 conv and 2 FC layers) and the number of channels could increase from 100 to 700. The image patch of each frame was passed through a stack of 3×3 conv layers. After the last max pooling layer, a 1×1 conv layer was used to increase the nonlinearity of the net without more computation and memory compared to other larger receptive fields. All conv layers are followed by the rectification nonlinearity (ReLU) and the stride was 1, while the stride of all max pooling layers was 2 with 3×3 window. Second, the BN operation was equipped for the outputs before nonlinearity in every convolutional layer to further improve the recognition rate with the realignment. All experiments were conducted based on our enhanced 4-gram language model, and other hyperparameters were selected based on the previous experiments, i.e., the frame length was set to 40, the frame shift was set to 3, and the state number was 5.

Table 9 The CER (in %) and model size comparison of DCNN and DNN under the same HMM framework on the evaluation set

System	DCNN	DNN
CER (%)	**4.07**	5.88
Model size(MB)	**107**	238

Table 10 The CER (in %) comparison of with or without BN/re_alignment for DCNN on the evaluation set

System	DCNN	BN_DCNN	ReFA_BN_DCNN
CER	4.07	3.75	**3.53**

Table 9 lists the CER and model size comparison of different NN architectures under the same HMM framework on the ICDAR 2013 competition set. The DNN structure was the same with the previous experiments. The state-level labels of GMM-HMM were the same for the DCNN and the DNN. Obviously, the DCNN model significantly outperformed the DNN model in terms of both CER and model size. The CER of DNN model was 5.88%, while the DCNN model could achieve a promising CER of 4.07%, yielding a relative CER reduction of 30.8% over the DNN model. And the model size of DCNN was 107 mega bytes (MB), which was less than half of that for DNN model.

We gradually improved the recognition rate on the basis of the DCNN. In Table 10, the only difference between the DCNN and the BN_DCNN was that we equipped the BN operation for the outputs before nonlinearity in every convolutional layer of the BN_DCNN. Similarly,the ReFA_BN_DCNN model performed one-pass realignment on top of the BN_DCNN model. The constant parameter of the BN ε in Eq. (7) was 10e−9 to keep numerical stability. Batch normalization could yield a relative CER reduction of 7.9% (from 4.07 to 3.75%), while realignment could achieve a relative CER reduction of 5.9% (from 3.75 to 3.53%).

6.5 Overall comparison

In Table 11, we give an overall comparison among our proposed approach, the best reported segmentation-free

Table 11 The CER (in %) comparison with the segmentation-free approach on the evaluation set

	DNN-HMM	DCNN-HMM	MDLSTM-RNN [12]
No LM	15.20	**10.34**	16.5
With LM	5.24	**3.53**	10.6

Table 12 The CER (in %) comparison with oversegmentation approaches on both CASIA-HWDB test set and ICDAR2013 competition set

	DCNN-HMM		Heterogeneous CNN [9]		CNN shape models + NNLM [10]	
	CASIA-HWDB	ICDAR2013	CASIA-HWDB	ICDAR2013	CASIA-HWDB	ICDAR2013
No LM	7.41	10.42	7.96	11.21	–	–
With LM	3.37	3.66	4.79	5.98	4.12	3.80

approach [12] on the evaluation set. Both 0-gram (No LM) and enhanced 4-gram LM (With LM) were considered. Several observations could be made. First, our proposed DNN-HMM and DCNN-HMM approaches were more effective than the previously best reported segmentation-free approach (MDLSTM-RNN). Second, when no LM was used, the DNN-HMM system with the best configuration in Table 8 yielded a relative CER reduction of 7.9% over MDLSTM-RNN approach demonstrating that our character modeling was more powerful and the training procedure was well designed to integrate the CE training, realignment, and sequence training to generate quite competitive performance .

In previous experiments, all the handwritten samples in training and test sets of CASIA-HWDB database were adopted. Meanwhile, the transcriptions of CASIA database were also used to generate the n-gram LM. To make a fair comparison with other approaches, we only used the training set of CASIA-HWDB to build the DCNN-HMM character model and removed the CASIA transcriptions to build the n-gram LM. Accordingly, as shown in Table 12, we conduct an overall comparison between our DCNN-HMM approach and other oversegmentation approaches on both CASIA-HWDB test set and ICDAR2013 competition set. Without using LM, the proposed DCNN-HMM approach outperformed the heterogeneous CNN approach in [9] on both CASIA-HWDB test set and ICDAR2013 competition set, indicating DCNN-HMM was a more powerful character model. For the case of using LM, DCNN-HMM achieved relative CER reductions of 18.2 and 3.7% over the best published oversegmentation approach in [10] on CASIA-HWDB test set and ICDAR2013 competition set, respectively.

7 Conclusion

In this study, we propose a hybrid neural network hidden Markov model approach for the offline HCTR. With the well-designed feature extraction, character modeling, and language modeling, the proposed approach yields a large performance gain over the previously best results. Considering the current implementation of sequence training is based on the lattices and the sequence training for DCNN-HMM requires too many computing and memory resources, we plan to tackle this problem by investigating the lattice-free MMI based sequence training method using a modified HMM structure [47] in the future work. Also adaptive technology will be explored for character modeling, and tied-state strategy will be considered to handle more character classes.

Acknowledgements This work was supported in part by the National Key R&D Program of China under Contract No. 2017YFB1002202, the National Natural Science Foundation of China under Grant Nos. 61671422 and U1613211, the Key Science and Technology Project of Anhui Province under Grant No. 17030901005, and MOE-Microsoft Key Laboratory of USTC.

References

1. Liu, C.-L., Jaeger, S., Nakagawa, M.: Online recognition of Chinese characters: the state-of-the-art. IEEE Trans. Pattern Anal. Mach. Intell. **26**(2), 198–213 (2004)
2. Fujisawa, H.: Forty years of research in character and document recognition—an industrial perspective. Pattern Recognit. **41**(8), 2435–2446 (2008)
3. Liu, C.-L., Yin, F., Wang, Q.-F., Wang, D.-H.: ICDAR 2011 Chinese handwriting recognition competition. In: Proceedings of the ICDAR, pp. 1464–1469 (2011)
4. Yin, F., Wang, Q.-F., Zhang, X.-Y., Liu, C.-L.: ICDAR 2013 Chinese handwriting recognition competition. In: Proceedings of the ICDAR, pp. 1464–1470 (2013)
5. Ding, X., Liu, H.: Segmentation-driven offline handwritten Chinese and Arabic script recognition. In: Proceedings of the Arabic and Chinese Handwriting, pp. 61–73 (2006)
6. Fu, Q., Ding, X.-Q., Liu, T., Jiang, Y., Ren, Z.: A novel segmentation and recognition algorithm for Chinese handwritten address character strings. In: Proceedings of the ICPR, pp. 974–977 (2006)
7. Li, N.-X., Jin, L.-W.: A Bayesian-based probabilistic model for unconstrained handwritten offline Chinese text line recognition. In: Proceedings of the IEEE SMC, pp. 3664–3668 (2010)
8. Wang, Q.-F., Yin, F., Liu, C.-L.: Handwritten Chinese text recognition by integrating multiple contexts. IEEE Trans. Pattern Anal. Mach. Intell. **34**(8), 1469–1481 (2012)

9. Wang, S., Chen, L., Xu, L., Fan, W., Sun, J., Naoi, S.: Deep knowledge training and heterogeneous CNN for handwritten Chinese text recognition. In: Proceedings of the ICFHR, pp. 84–89 (2016)
10. Wu, Y.-C., Yin, F., Liu, C.-L.: Improving handwritten Chinese text recognition using neural network language models and convolutional neural network shape models. Pattern Recognit. **65**, 251–264 (2017)
11. Su, T.-H., Zhang, T.-W., Guan, D.-J., Huang, H.-J.: Off-line recognition of realistic Chinese handwriting using segmentation-free strategy. Pattern Recognit. **42**(1), 167–182 (2009)
12. Messina, R., Louradour, J.: Segmentation-free handwritten Chinese text recognition with LSTM-RNN. In: Proceedings of the ICDAR, pp. 171–175 (2015)
13. Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H., Schmidhuber, J.: A novel connectionist system for improved unconstrained handwriting recognition. IEEE Trans. Pattern Anal. Mach. Intell. **31**(5), 855–868 (2009)
14. Graves, A., Fernandez, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the ICML, pp. 369–376 (2006)
15. Suryani, D., Doetsch, P., Ney, H.: On the benefits of convolutional neural network combinations in offline handwriting recognition. In: Proceedings of the ICFHR (2016)
16. Bunke, H., Bengio, S., Vinciarelli, A.: Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. IEEE Trans. Pattern Anal. Mach. Intell. **26**(6), 709–720 (2004)
17. Guo, Q., Wang, F.-L., Lei, J., Tu, D., Li, G.-H.: Convolutional feature learning and hybrid CNN-HMM for scene number recognition. Neurocomputing **184**, 78–90 (2016)
18. Kozielski, M., Doetsch, P., Ney, H.: Improvements in RWTHs system for off-line handwriting recognition. In: Proceedings of the ICDAR, pp. 935–939 (2013)
19. Chen, K., Yan, Z.J., Huo, Q.: A context-sensitive-chunk BPTT approach to training deep LSTM/BLSTM recurrent neural networks for offline handwriting recognition. In: Proceedings of the ICDAR, pp. 411–415 (2015)
20. Du, J., Wang, Z.-R., Zhai, J.-F., Hu, J.-S.: Deep neural network based hidden Markov model for offline handwritten Chinese text recognition. In: Proceedings of the ICPR (2016)
21. Wang, Z.-R., Du, J., Hu, J.-S., Hu, Y.-L.: Deep convolutional neural network based hidden Markov model for offline handwritten Chinese text recognition. In: Proceedings of the ACPR (2017)
22. Dahl, G., Yu, D., Deng, L., Acero, A.: Context-dependent pre-trained deep neural networks for large vocabulary speech recognition. IEEE Trans. Audio Speech Lang. Process. **20**(1), 30–42 (2012)
23. Veselý, K., Ghoshal, A., Burget, L., Povey, D.: Sequence-discriminative training of deep neural networks. In: Proceedings of the Interspeech, pp. 2345–2349 (2013)
24. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167 (2015)
25. Jelinek, F.: The development of an experimental discrete dictation recognizer. Proc. IEEE **73**(11), 1616–1624 (1985)
26. Mohri, M., Pereira, F., Riley, M.: Weighted finite-state transducers in speech recognition. Comput. Speech Lang. **20**(1), 69–88 (2002)
27. Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., Mohri, M.: OpenFst: a general and efficient weighted finite-state transducer library. In: Proceedings of the CIAA, pp. 11–23 (2007)
28. Liu, C.-L., Nakashima, K., Sako, H., Fujisawa, H.: Handwritten digit recognition: benchmarking of state-of-the-art techniques. Pattern Recognit. **36**(10), 2271–2285 (2003)
29. Liu, C.-L.: Normalization-cooperated gradient feature extraction for handwritten character recognition. IEEE Trans. Pattern Anal. Mach. Intell. **29**(8), 1465–1469 (2007)
30. Bai, Z.-L., Huo, Q.: A study on the use of 8-directional features for online handwritten Chinese character recognition. In: Proceedings of the ICDAR, pp. 262–266 (2005)
31. Rencher, A.C.: Methods of Multivariate Analysis. Wiley, New York (2002)
32. Baum, L.E., Eagon, J.A.: An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. Bull. Am. Math. Soc. **73**, 360–363 (1967)
33. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE **77**(2), 257–286 (1989)
34. Juang, B.H., Levinson, S., Sondhi, M.: Maximum likelihood estimation for multivariate mixture observations of Markov chains. IEEE Trans. Inf. Theory **32**(2), 307–309 (1986)
35. Young, S., et al.: The HTK Book (Revised for HTK Version 3.4.1). Cambridge University, Cambridge (2009)
36. Povey, D., Ghoshal, A. et al.: The kaldi speech recognition toolkit. In: Proceedings of the ASRU (2011)
37. Bahl, L.R., Brown, P.F., de Souza, P.V., Mercer, R.L.: Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In: Proceedings of the ICASSP, pp. 49–52 (1986)
38. Povey, D.: Discriminative training for large vocabulary speech recognition. Ph.D. dissertation, University of Cambridge, Cambridge, UK (2003)
39. Povey, D., Kingsbury, B.: Evaluation of proposed modifications to MPE for large scale discriminative training. In: Proceedings of the ICASSP, pp. IV-321–IV-324 (2007)
40. Hubel, D.H., Wiesel, T.N.: Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. J. Physiol. **160**, 106–154 (1962)
41. Kneser, R., Ney, H.: Improved backing-off for m-gram language modeling. In: Proceedings of the ICASSP, pp. 181–184 (1995)
42. Katz, S.: Estimation of probabilities from sparse data for the language model component of a speech recognizer. IEEE Trans. Acoust. Speech Signal Process. **35**(3), 400–401 (1987)
43. Stolcke, A.: SRILM: an extensible language modeling toolkit. In: Proceedings of the ICSLP, pp. 901–904 (2002)
44. Liu, C.-L., Yin, F., Wang, D.-H., Wang, Q.-F.: CASIA online and offline Chinese handwriting databases. In: Proceedings of the ICDAR, pp. 37–41 (2011)
45. Liu, C.-L., Yin, F., Wang, D.-H., Wang, Q.-F.: Online and offline handwritten Chinese character recognition: benchmarking on new databases. Pattern Recognit. **46**(1), 155–162 (2013)
46. Jia, Y.-Q., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: convolutional architecture for fast feature embedding. Preprint arXiv:1408.5093 (2014)
47. Povey, D., Peddinti, V. et al.: Purely sequence-trained neural networks for ASR based on lattice-free MMI. In: Proceedings of the Interspeech, pp. 2751–2755 (2016)