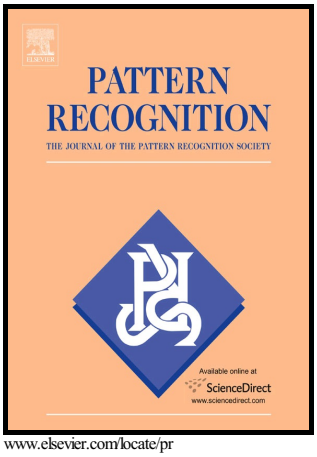
Improving handwritten chinese text recognition using neural network language models and convolutional neural network shape models

Yi-Chao Wu, Fei Yin, Cheng-Lin Liu



www.elsevier.com/locate/pr

PII: S0031-3203(16)30447-2
DOI: http://dx.doi.org/10.1016/j.patcog.2016.12.026
Reference: PR5998

To appear in: *Pattern Recognition*

Received date: 26 February 2016
Revised date: 23 December 2016
Accepted date: 24 December 2016

Cite this article as: Yi-Chao Wu, Fei Yin and Cheng-Lin Liu, Improving handwritten chinese text recognition using neural network language models and convolutional neural network shape models, *Pattern Recognition* http://dx.doi.org/10.1016/j.patcog.2016.12.026

Improving Handwritten Chinese Text Recognition Using Neural Network Language Models and Convolutional Neural Network Shape Models

Yi-Chao Wu[a], Fei Yin[a], Cheng-Lin Liu[a,b,c,*]

[a] *National Laboratory of Pattern Recognition (NLPR)*
Institute of Institute of Automation of Chinese Academy of Sciences
Beijing 100190, China
[b] *University of Chinese Academy of Sciences, Beijing, China*
[c] *CAS Center for Excellence in Brain Science and Intelligence Technology, Beijin, China*

Abstract

Handwritten Chinese text recognition based on over-segmentation and path search integrating multiple contexts has been demonstrated successful, wherein the language model (LM) and character shape models play important roles. Although back-off n-gram LMs (BLMs) have been used dominantly for decades, they suffer from the data sparseness problem, especially for high-order LMs. Recently, neural network LMs (NNLMs) have been applied to handwriting recognition with superiority to BLMs. With the aim of improving Chinese handwriting recognition, this paper evaluates the effects of two types of character-level NNLMs, namely, feedforward neural network LMs (FNNLMs) and recurrent neural network LMs (RNNLMs). Both FNNLMs and RNNLMs are also combined with BLMs to construct hybrid LMs. For fair comparison with BLMs and a state-of-the-art system, we evaluate in a system with the same character over-segmentation and classification techniques as before, and compare various LMs using a small text corpus used before. Experimental results on the Chinese handwriting database CASIA-HWDB validate that NNLMs improve the recognition performance, and hybrid RNNLMs outperform the other LMs. To report a new benchmark, we also evaluate selected LMs on a large corpus, and replace the baseline character

*Corresponding author
Email addresses: yichao.wu@nlpr.ia.ac.cn (Yi-Chao Wu), fyin@nlpr.ia.ac.cn (Fei Yin), liucl@nlpr.ia.ac.cn (Cheng-Lin Liu)

classifier, over-segmentation, and geometric context models with convolutional neural network (CNN) based models. The performance on both the CASIA-HWDB and the ICDAR-2013 competition dataset are improved significantly. On the CASIA-HWDB test set, the character-level accurate rate (AR) and correct rate (CR) achieve 95.88% and 95.95%, respectively.

Keywords: Handwritten Chinese text recognition, Feedforward neural network language model, Recurrent neural network language model, Hybrid language model, Convolutional neural network shape models

1. Introduction

For the past forty years, the field of handwritten Chinese text recognition (HCTR) has observed tremendous progresses [1, 2]. However, it remains a challenging problem due to the diversity of writing styles, the character segmentation difficulty, large character set and unconstrained language domain. The recognition approach based on over-segmentation by integrating character classifier, geometric and linguistic context models has been demonstrated successful in handwritten text recognition [3], among which both the linguistic context model (i.e., language model) and the character shape models are of great importance.

Statistical language models, which give the prior probability of a sequence of characters or words, play an important role in many applications such as character and speech recognition, machine translation and information retrieval, etc. Although back-off N-gram language models (BLMs) were proposed over twenty years ago [4, 5] and have been used in handwritten text recognition for more than ten years, they are still considered as a favorable choice and have performed superiorly for decades. BLMs have been widely applied in a vast variety of text recognition systems [3, 6–13], and have boosted the recognition performance substantially.

Generally, higher order language models can capture longer context patterns so as to estimate the sequence probability more accurately. Carpenter [14] found that the performance of character N-gram can be significantly improved until 8-gram, given sufficient training samples. However, traditional BLMs suffer from the data sparse-

ness problem, as the number of parameters increases exponentially with the length of the context (i.e., the curse of dimensionality), preventing these models from estimating context stably. Recently, a new type of language model called neural network language model (NNLM) has been proposed to address the data sparseness based on a continuous representation of words [15], and achieves great perplexity reduction compared with BLMs. Since then, NNLMs have been successfully used in speech recognition [16, 17], machine translation [18, 19], and handwriting recognition [20, 21]. Meanwhile, many extensions of NNLMs and related algorithms have been proposed, with the aim of improving the model performance [17, 22–25] or reducing the time complexity [26–29]. Particularly, the previous work has focused on either feedforward NNLMs (FNNLMs) [15, 16, 18–21] or recurrent NNLMs (RNNLMs) [17, 28, 30, 31]. Nevertheless, to the best of our knowledge, except for our previous work on FNNLMs [21], there is no systematic evaluation of NNLMs for over-segmentation based text recognition systems.

Apart from the language model, character classifier [32], over-segmentation [21, 33, 34] and geometric context models [35] (called shape models generally in this paper) are also important to the text recognition performance. CNN based classifiers for Chinese characters have reported superior performance in ICDAR 2013 competition [36], where CNNs reported much higher accuracies than traditional classifiers. Using CNNs, the handwriting recognition community has reported many useful and important achievements [37–39] to improve the recognition accuracy. Recently, by integrating the traditional normalization-cooperated direction-decomposed feature map (directMap) with the deep CNN, Zhang et al. [40] obtained new highest accuracies for both online and offline sessions on the ICDAR-2013 competition database. For over-segmentation, there have been many algorithms to deal with touching characters, but most of them are based on heuristic rules [33, 34, 41, 42], which make it very difficult to generalize from one application to another. A few learning based techniques have been explored [10, 43, 44], however, only the method in [44] was successfully applied in HCTR, and is only suitable for the single-touching situation. As for the geometric context models, although many researchers proved it can improve the recognition accuracy [3, 9, 35, 45, 46], there has been no work using deep learning based geometric

models.

In this paper, we evaluate the effects of two types of character-level NNLMs, namely, FNNLMs and RNNLMs, with the aim of improving Chinese handwriting recognition. Both FNNLMs and RNNLMs are also combined with BLMs to construct hybrid LMs. For fair comparison with BLMs and a state-of-the-art system, we evaluate in a system with the same character over-segmentation and classification techniques as before, and compare various LMs using a small text corpus that were used by a previous system. In experiments on the Chinese handwriting database CASIA-HWDB, the comparison of a number of variations of LMs shows that the NNLMs improve the recognition performance, and hybrid RNNLMs outperform the other LMs. To provide a new benchmark, we then evaluate selected LMs on a large corpus. Also, we replace all the baseline character classifier, over-segmentation algorithm, and geometric context models with CNN-based models in the system for further improving the accuracy of HCTR. By doing these, the performance on both the CASIA-HWDB and the ICDAR-2013 competition dataset are improved significantly. Specifically, on the CASIA-HWDB dataset, the character-level accurate rate (AR) and correct rate (CR) achieve 95.88% and 95.95%, respectively compared to the previous results of 90.75% AR and 91.39% CR (with candidate character augmentation) [3], 91.73% AR and 92.37% CR (with language model adaptation) [12], 95.21% AR and 96.28% CR (with CNN character classifier) [32].

The major contributions of this work are in three respects. First, we perform a comprehensive evaluation of NNLMs in handwritten Chinese text recognition and propose hybrid NNLMs to improve the performance. Second, we apply CNNs to over-segmentation and geometric context modeling in addition to character recognition. Third, by training NNLMs on large corpus and integrating CNN shape models, we achieve new state-of-the-art performance on standard datasets. In addition, we analyze the upper bound of performance of the text recognition system by calculating the lattice error rate, which shows the potential of improvement in the future.

The rest of this paper is organized as follows: Section 2 reviews some related works; Section 3 gives an overview of the handwritten Chinese text recognition system; Section 4 describes the FNNLMs and RNNLMs, as well as techniques for accelerat-

ing them; Section 5 presents the CNN based models, including character classifier, over-segmentation algorithm, and geometric context models; Section 6 presents experimental results, and Section 7 offers concluding remarks.

2. Related works

The neural network architecture has a strong impact on the performance of NNLMs, and comparative studies have been conducted by some researchers [28, 47–51]. Mikolov et al. [28] gave an empirical comparison between RNNLMs and FNNLMs on two corpora, and found that simple RNNLMs outperformed the standard FNNLMs in terms of perplexity (PPL) on both the Penn Tree Bank and the Switchboard corpus. Mikolov et al. [47] presented PPL results obtained with several advanced language modeling techniques, including some types of NNLMs, and the results demonstrated the superiority of RNNLMs. However, neither of them made comparisons in practical speech or text recognition systems, where the inputs are often contaminated by noises. Sundermeyer et al. [48] compared recurrent Long Short-Term Memory (LSTM) NNLMs, which can be considered as a variant of RNNLMs, with FNNLMs on a well-tuned French speech recognition task. Their results showed that LSTM-NNLMs achieved lower PPL than FNNLMs, and also reduced the word error rate (WER). This work was extended in [49], to compare BLMs with FNNLMs, RNNLMs, and LSTM-NNLMs on two large-vocabulary speech recognition tasks. The results showed that both LSTM-NNLMs and RNNLMs outperformed FNNLMs in terms of PPL and WER. Arisoy et al. [50] even compared deep FNNLMs with RNNLMs, and there seemed to be no evident improvements in PPL or WER for deep models. Almost all the previous works validate the superiority of RNNLMs to FNNLMs, except for [51], where RNNLMs was outperformed by a 10-gram FNNLM in PPL. However, the final result of BLEU on a large scale English to French translation task was identical for both network structures.

In the field of handwritten text recognition, only a few people have investigated the potential of NNLMs [20, 21, 52]. Zamora-Martínez et al. [20] integrated FNNLMs in the decoding process of three state-of-the-art systems for English handwriting recognition. Experimental results demonstrated that consistent WER reductions can be

achieved by FNNLMs when compared with BLMs on the three tested systems. Li et al. [53] applied RNNLMs to the n-best rescoring stage of the state-of-the-art BBN Byblos OCR (optical character recognition) system for handwriting recognition and achieved significant improvement. Our work [21] was the first to investigate the effects of NNLMs in handwritten Chinese text recognition. The recognition results on the CASIA-HWDB database [54] showed that simplified NNLMs and BLMs of the same order performed comparably, and hybrid models constructed by interpolating NNLMs and BLMs improved the recognition performance significantly. However, to our knowledge, in either English or Chinese handwritten text recognition, NNLMs have not been evaluated systematically.

On the other hand, many works concerning character shape model and geometric context have been proposed for handwritten text recognition. For character classification, traditional methods usually involve character normalization, feature extraction, and classifier design, which have been reviewed in [55, 56]. Nowadays, the solution of handwritten Chinese character recognition (HCCR) has been changed from traditional methods to CNNs because of their superior performances. The first reported successful use of CNN for HCCR is the multi-column deep neural network [37]. Alternately trained relaxation convolutional neural network was proposed by [38] for offline HCCR. The methods of [39, 40], by combining traditional feature extraction methods such as Gabor and gradient feature maps with deep CNN, also obtained very high recognition accuracies. Learning based over-segmentation has been explored for decades, and has achieved great success in separating characters with high recall rate [10, 43, 44]. The method referred to as GraySeg [10] combines the output of a sliding window classifier and boundaries of connected components (CCs) for over-segmentation, and has led to superior text recognition performance on public benchmark datasets. Xu et al. [44] proposed an effective over-segmentation method with learning-based filtering using geometric features for single-touching Chinese handwriting. The geometric context also plays an important role in character string recognition [3, 9, 35, 45, 46]. Zhou et al. [9, 35] elaborated the geometric context models into unary and binary, character class dependent and class-independent models in online handwriting recognition. Wu et al. [46] demonstrated that geometric context is bene-

ficial to handwritten numeral string recognition, and they also proposed an improved binary geometric model to further improve the system performance. Yin et al. [57] elaborated the geometric context models for offline handwriting and applied to transcript mapping of handwritten Chinese documents.

3. System overview

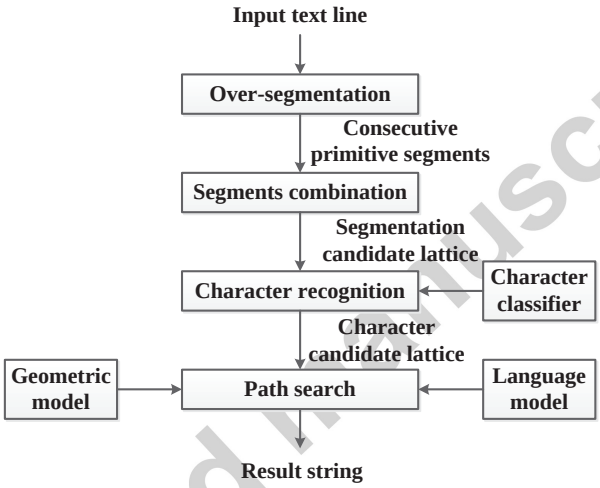


Figure 1: System diagram of handwritten Chinese text line recognition.

Our system is based on the integrated segmentation-and-recognition framework, which typically consists the steps of over-segmentation of a text line image, construction of the segmentation-recognition candidate lattice, and path search in the lattice with context fusion. The diagram of our system is shown in Fig. 1, and the tasks of document image pre-processing and text line segmentation are assumed to have been accomplished externally.

First, the input text line image is over-segmented into a sequence of primitive image segments by connected component analysis and touching pattern splitting [1] [3, 33] (Fig.

[1] We first use existing over-segmentation technique in evaluating NNLMs, and later apply CNN to over-segmentation for higher recognition performance.

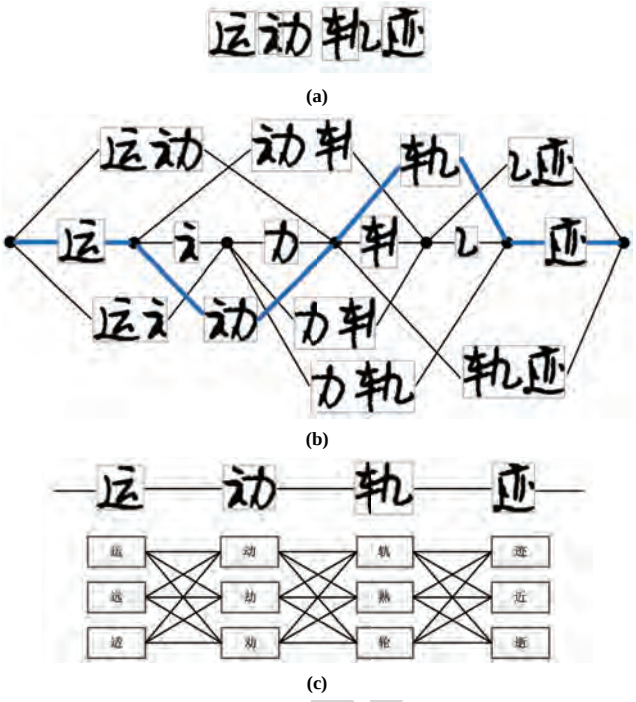


Figure 2: (a) Over-segmentation of a text line; (b) Segmentation candidate of (a); (c) Character candidate lattice of the thick path in (b).

2(a)), so that each segment is a character or a part of a character. Then, one or more consecutive segments are combined to generate candidate character patterns, forming a segmentation candidate lattice as shown in Fig. 2(b), and each path in this lattice is called a candidate segmentation path. Each candidate pattern is classified to assign a number of candidate character classes using a character classifier, and all the candidate patterns in a candidate segmentation path form a character candidate lattice, which is shown in Fig. 2(c). All of these character candidate lattices are merged to construct the segmentation-recognition lattice of the input text line, and each path in this lattice is constructed by a character sequence paired with a candidate pattern sequence, which is called a candidate segmentation-recognition path. The rest of the task is to evaluate each path by fusing multiple contexts and to search the optimal path with minimum cost or maximum score to obtain the segmentation and recognition result.

We denote a sequence of candidate character patterns as $X = x_1...x_m$. Each candi-

date character is assigned candidate class (denoted as c_i) by a character classifier, and then the result of text line recognition is a character string $C = c_1...c_m$. In this work, we formulate the task of string recognition from Bayesian decision view, and adopt the path evaluation criterion presented in [3] which integrates the character classification score, geometric context [57] and linguistic context. For saving space, we give the criterion directly below, and more details can be found in [3].

Denote the character classifier output of candidate class c_i for the ith character pattern x_i as $P(c_i|x_i)$. The linguistic context is denoted as $P(c_i|h_i)$, where h_i denotes the history of c_i (see Section 4). The geometric context models give the unary class-dependent geometric (**ucg**) score, unary class-independent geometric (**uig**) score, binary class-dependent geometric (**bcg**) score and binary class-independent geometric (**big**) score, denoted as $P(c_i|g_i^{ucg})$, $P(z_i^p = 1|g_i^{uig})$, $P(c_{i-1}, c_i|g_i^{bcg})$, and $P(z_i^g = 1|g_i^{big})$, respectively, where g_i denotes corresponding geometric features, and the output scores are given by geometric models classifying on features extracted. We obtain a log-likelihood function $f(X, C)$ for the segmentation-recognition path:

$$f(X,C) = \sum_{i=1}^{m}(w_i \log P(c_i|x_i) + \lambda_1 \log P(c_i|g_i^{ucg}) + \lambda_2 \log P(z_i^p = 1|g_i^{uig}) + \lambda_3 \log P(c_{i-1}, c_i|g_i^{bcg}) + \lambda_4 \log P(z_i^g = 1|g_i^{big}) + \lambda_5 \log P(c_i|h_i)), \quad (1)$$

where w_i is the word insertion penalty used to overcome the bias to short strings, for which we utilize the term of Weighting with Character Pattern Width (WCW) [3], λ_1-λ_5 are the weights to balance the effects of different models and are optimized with Maximum Character Accuracy (MCA) criterion [3]. Via confidence transformation (transforming classifier output scores to probabilities), the six models, namely, one character classifier, four geometric models and one character linguistic model, are combined to evaluate the segmentation paths. As for path search, a refined frame-synchronous beam search algorithm [3] is employed to find the optimal paths in two steps: first retain a limited number of partial paths with maximum scores at each frame, and then find the globally optimal path in the second step.

4. Neural network language models

To overcome the data sparseness problem of traditional BLMs, we introduce two types of NNLMs including FNNLMs and RNNLMs in this section.

If the sequence C contains m characters, $P(C)$ can be decomposed as:

$$p(C) = \prod_{i=1}^{m} p(c_i | c_1^{i-1}), \tag{2}$$

where $c_1^{i-1} =< c_1, ..., c_{i-1} >$ denotes the history of character c_i. For an N-gram model, it only considers the $N-1$ history characters in (2):

$$p(C) = \prod_{i=1}^{m} p(c_i | c_{i-N+1}^{i-1}) = \prod_{i=1}^{m} p(c_i | h_i), \tag{3}$$

where $h_i = c_{i-N+1}^{i-1} =< c_{i-N+1}, ..., c_{i-1} >$ (h_1 is null). Although FNNLMs can be trained with larger context sizes than BLMs, it is intrinsically an N-gram LMs as well. However, RNNLMs can get rid of limited context size and capture unbounded context patterns in theory. Therefore, we have $h_i = c_1^{i-1}$ in this case.

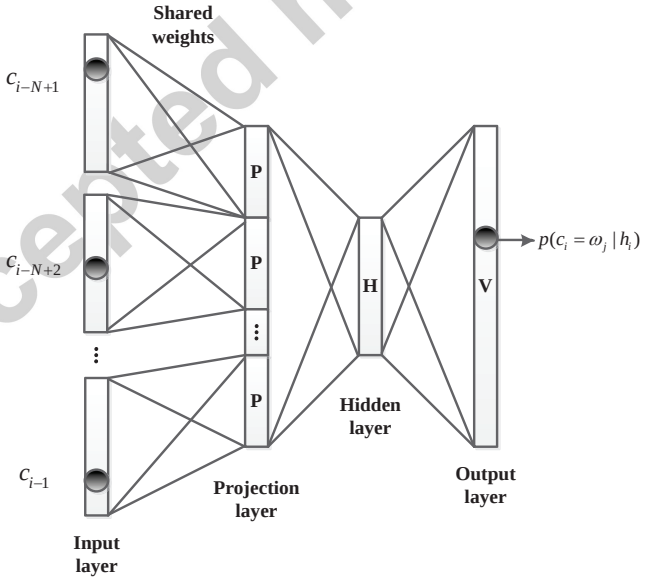


Figure 3: Architecture of FNNLM with one hidden layer. P is the size of one projection, and H, V are the sizes of the hidden and output layer, respectively.

4.1. Feedforward neural network language models

In FNNLMs, history characters (or words for English texts)[2] are projected into a continuous space to perform an implicit smoothing and estimate the probability of a sequence. Both the projection and estimation can be jointly performed by a multi-layer neural network. The original FNNLM model was proposed by Bengio [15] to attack the curse of dimensionality, and the basic architecture with one hidden layer is depicted in Fig. 3.

The input of the N-gram FNNLM is formed by concatenating the information of $N-1$ history characters h_i, while the outputs are the posterior probabilities of all the characters in the vocabulary:

$$p(c_i = \omega_j | h_i), \qquad j = 1, \ldots, V, \tag{4}$$

where V is the size of the vocabulary, and ω_j denotes a character class in the vocabulary. The network functions as follows:

(1) Each of the previous $N-1$ input characters is initially encoded as a vector with length V using the 1-of-V scheme.

(2) Then, each 1-of-V representation of character is projected to a lower dimensional vector denoted as $\boldsymbol{r}$ in a continuous space. In fact, each column of the $P \times V$ dimensional projection matrix corresponds to a distributed representation, and all the weights of the projection layer are shared.

(3) After step 2, if we denote the weights between the projection layer and the hidden layer as $\boldsymbol{W}_{P,H}$ whose dimension should be $H \times ((N-1) \times P)$ using the column-major form, the $N-1$ history characters' distributed representations as $\boldsymbol{R} = [\boldsymbol{r}_{i-N+1}^T, ..., \boldsymbol{r}_{i-1}^T]^T$, then the hidden layer outputs $\boldsymbol{S}$ can be computed as:

$$\boldsymbol{S} = \tanh(\boldsymbol{W}_{P,H} * \boldsymbol{R}), \tag{5}$$

where $\tanh(\cdot)$ is the hyperbolic tangent activation function performed element wise. If there are multiple hidden layers, the same processing of Eq. (5) applies to the succeeding hidden layer with the former hidden layer outputs as inputs.

[2] We take characters instead of words as the elements (grams) in Chinese vocabulary, but may term characters and words interchangeably when referring to previous works.

(4) Finally, the prediction of all the characters in the vocabulary can be calculated by

$$\boldsymbol{M} = \boldsymbol{W}_{H,O} * \boldsymbol{S}, \tag{6}$$

$$\boldsymbol{O} = \exp(\boldsymbol{M}) / \sum_{i=1}^{V} \exp\left(m_i\right), \tag{7}$$

where $\boldsymbol{W}_{H,O}$ is the $V \times H$ dimensional weight matrix of the output layer, $\boldsymbol{M}$ is the vector of the activation values calculated before softmax normalization, m_i is the ith element of $\boldsymbol{M}$. The $\exp(\cdot)$ function as well as the division function are performed element wise.

The above formulas have absorbed the bias items into the weight parameters for the sake of illustration simplicity [58]. After all the above operations, the jth component of $\boldsymbol{O}$, denoted as o_j, corresponds to the probability $p(c_i = \omega_j | h_i)$. The standard back-propagation algorithm is used in training to minimize the regularized cross-entropy criterion:

$$E = -\sum_{j=1}^{V} t_j \log o_j + \beta(|\boldsymbol{W}_{P,H}|_2^2 + |\boldsymbol{W}_{H,O}|_2^2), \tag{8}$$

where t_j is the desired output, which is 1 for the next character in the training sentence, and 0 for the others.

4.2. Recurrent neural network language models

RNNLMs were firstly proposed in [17]. Its architecture (Fig. 4) is similar to that of FNNLMs, except that the hidden layer involves recurrent connections. The RNNLM embeds word/character representation projection as well, and there are mainly three stages for estimation:

(1) The input $\boldsymbol{R}(t)$ of the time step t is firstly formed by concatenating two parts: vector $\boldsymbol{x}(t-1)$ representing the previous word c_{i-1} by 1-of-V coding, and the previous hidden layer output $\boldsymbol{S}(t-1)$, expressed as:

$$\boldsymbol{R}(t) = [\boldsymbol{x}(t-1)^T \boldsymbol{S}(t-1)^T]^T. \tag{9}$$

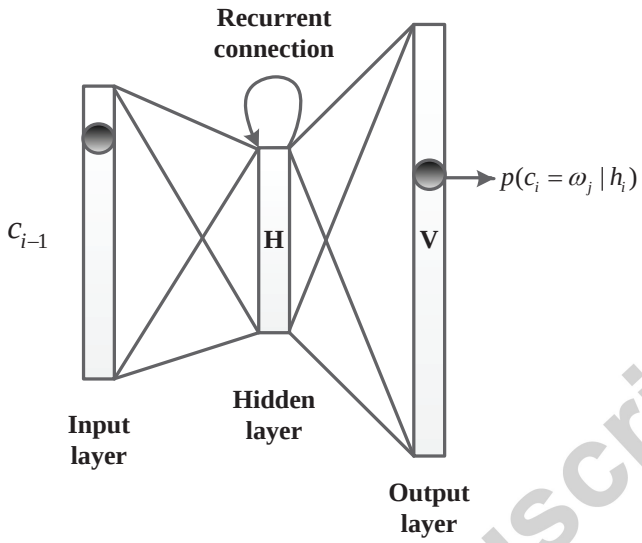


Figure 4: Architecture of RNNLM. H, V are the sizes of the hidden layer and the output layer, respectively.

(2) The input $\boldsymbol{R}(t)$ is then separately projected to a continuous vector $\boldsymbol{S}(t)$, which is also the hidden layer for the next time step:

$$\boldsymbol{S}(t) = \text{sigm}(\boldsymbol{W}_{I,H} * \boldsymbol{x}(t-1) + \boldsymbol{W}_{H,H} * \boldsymbol{S}(t-1)), \tag{10}$$

where $\text{sigm}(\cdot)$ is the sigmoid activation function performed element wise, $\boldsymbol{W}_{I,H}$ and $\boldsymbol{W}_{H,H}$ are $H \times V$ projection and $H \times H$ recurrent weight matrices, respectively.

(3) The probabilities of all the words in the vocabulary are estimated in the same way as the 4th step of FNNLMs in Section 4.1.

The RNNLM is trained by minimizing a regularized cross-entropy criterion similar to that in Eq. (8). However, the recurrent weights are optimized using the back-propagation through time (BPTT) algorithm [28], and the truncated BPTT is used to prevent the gradient vanishing or explosion problems.

The main difference between FNNLMs and RNNLMs lies in the representation of the history. For FNNLMs, the history is restricted to limited previous characters; while for RNNLMs, because of the recurrent connection, the hidden layer represents

the whole history of text theoretically. In this way, RNNLMs can efficiently explore the context of longer sequence than FNNLMs.

4.2.1. RNN maximum entropy (RNNME) models

It was observed that for use with larger corpus, the architecture of RNNLMs should have more hidden units [30], otherwise, the performance can be even inferior to that of BLMs. However, the increase of hidden layer size also increases the computational complexity. To overcome this problem, Mikolov et al. combined RNNLMs with maximum entropy models [30]. The resulting model, called RNNME, can be trained jointly with BPTT. The RNNME model yielded promising performance with relatively small hidden layer sizes.

The maximum entropy model can be seen as a weight matrix that directly connects the input layer and the output layer in neural networks. When using N-gram features [59], the direct connection part can offer complementarity to RNNLMs, so that RNNME can achieve superior performance with relatively simple structures. Furthermore, it is natural to improve the efficiency of RNNME using hashing, and the RNNME can be viewed as a pruned model with a small hash array.

4.3. Hybrid language models

For use with large vocabulary tasks, it is a common practice to linearly interpolate an NNLM with a standard BLM for further improvement [19]. In such hybrid language models (HLMs), the interpolation weights are usually estimated by minimizing the perplexity (PPL) on a development dataset.

To overcome the high computational complexity, NNLMs are usually simplified with simple structures or approximation techniques. The simplified models are then combined with BLMs to give hybrid models. Due to the great complementarity of NNLMs to BLMs [15, 16, 47, 60], it was observed that even NNLMs with moderate performance can considerably improve the performance of HLMs [21]. This can be attributed to the fact that NNLMs and BLMs learn very different distributions [29]. We will show experimental results to validate this kind of complementarity in Section 6.

4.4. Acceleration

NNLMs suffer from high computational complexity in both training and testing, due to the layer-by-layer matrix computation, unlike BLMs that calculate and retrieve probabilities directly. Considering that the complexity of NNLMs is basically proportional to $O(|V|)$ [15], i.e., the softmax operation of output layer dominates the processing time, there have been two mainstream techniques for acceleration: short-list and output factorization, which are outlined as follows.

4.4.1. Short-list

Inspired from the work of Bengio et al. [15], Schwenk et al. proposed the short-list method [19] and successfully applied it to lattice rescoring of speech recognition systems. This method chooses the s ($s \ll V$) most frequent words, called as a short-list, to reduce the output units of the neural network. The output probabilities are then calculated as:

$$\hat{P}(c_i|h_i) = \begin{cases} \hat{P}_N(c_i|h_i, L) \cdot P_B(h_i|L), & \text{if } c_i \in \text{short-list} \\ \hat{P}_B(c_i|h_i), & \text{otherwise} \end{cases} \tag{11}$$

where $\hat{P}_N$ denotes the probability of words in the short-list calculated by NNLMs, $\hat{P}_B$ is the probability given by standard BLMs, the random variable L defines the event that the word to be predicted is in the short-list, and $P_B(h_i|L)$ is given by:

$$P_B(h_i|L) = \sum_{c_i \in \text{short-list}} \hat{P}_B(c_i|h_i). \tag{12}$$

For further simplification of the short-list method [19], an extra output unit is added for all the words that are not in the short-list, and its probability is learned by the neural network. We simply assume that this probability is sufficiently close to the probability mass reserved by the BLM. Thus, (11) can be modified as:

$$\hat{P}(c_i|h_i) = \begin{cases} \hat{P}_N(c_i|h_i), & \text{if} \quad c_i \in \text{short-list} \\ \hat{P}_B(c_i|h_i), & \text{otherwise} \end{cases} \tag{13}$$

It has been observed that there is no significant difference between the methods with and without renormalization [19]. Therefore, we can easily see that the time complexity is approximately reduced to $O(|s|)$ by the short-list method.

4.4.2. Output factorization

The idea of output factorization was originated from [26] (based on [61] in the context of maximum entropy models), where a binary hierarchical clustering constrained by the prior knowledge is used to decompose the output layer hierarchically, so that the complexity is reduced to $\log_2 |V|$. Since the construction of the hierarchical structure is not trivial, class-based models are usually adopted in practice. In this model, all the words are categorized into a smaller number of classes, and the word probability at the output layer can be factorized as

$$p(c_i|h_i) = p(\text{class}(c_i)|h_i) * p(c_i|\text{class}(c_i), h_i). \quad (14)$$

This shows that we can normalize the class distribution and the class-specific word distribution separately. The class-based output factorization has been observed to bring about 15 times speedup against models which uses full vocabulary of size 10K, and was said to be a more promising approach than the short-list method [28].

The methods for constructing word classes have been investigated in [62]. Although frequency-based categorization does not achieve better performance compared with likelihood-based categorization [63, 64] in terms of PPL, it has great advantage in speed. Therefore, we use frequency-based categorization for a better tradeoff between performance and speed. Specifically, we employ a modified algorithm which groups words based on square-root of the frequency instead of the frequency itself [65].

5. Convolutional neural network shape models

With the impact of the success of deep learning [66, 67] in different domains, we consider altering the modules of HCTR framework, namely, character classifier, over segmentation, and geometric context models from traditional methods to convolutional neural network (CNN) [68] based models. These models take character or text images as input, and so, are called shape models in general.

5.1. Character classifier

In this work, we build a 15-layer CNN as the character classifier as shown in Table 1, which is similar to the one proposed in [40]. Similar to the domain-specific

Table 1: CNN character classifier configuration. The first row is the bottom layer. k, s and p stand for kernel size, stride and padding size, respectively.

Type	Configurations
input	$9 \times 32 \times 32$ extended directMaps
Convolution	#maps: 50, k: 3×3, s:1, p:1, dropout: 0.0
Convolution	#maps: 100, k: 3×3, s:1, p:1, dropout: 0.1
Convolution	#maps: 100, k: 3×3, s:1, p:1, dropout: 0.1
MaxPooling	Window: 2×2, s: 2
Convolution	#maps: 150, k: 3×3, s:1, p:1, dropout: 0.2
Convolution	#maps: 200, k: 3×3, s:1, p:1, dropout: 0.2
Convolution	#maps: 200, k: 3×3, s:1, p:1, dropout: 0.2
MaxPooling	Window: 2×2, s: 2
Convolution	#maps: 250, k: 3×3, s:1, p:1, dropout: 0.3
Convolution	#maps: 300, k: 3×3, s:1, p:1, dropout: 0.3
Convolution	#maps: 300, k: 3×3, s:1, p:1, dropout: 0.3
MaxPooling	Window: 2×2, s: 2
Convolution	#maps: 350, k: 3×3, s:1, p:1, dropout: 0.4
Convolution	#maps: 400, k: 3×3, s:1, p:1, dropout: 0.4
Convolution	#maps: 400, k: 3×3, s:1, p:1, dropout: 0.4
MaxPooling	Window: 2×2, s: 2
Full connection	#hidden units: 1600 , dropout: 0.5
Full connection	#hidden units: 200, dropout: 0.0
Softmax	#units: 7357

knowledge incorporated in CNN [69], we extract eight 32×32 directMaps using line density projection interpolation normalization [70], as used in [40] as well. Besides the directMaps, we resize the original character image to 32×32 while keeping the aspect ratio as an extra input feature map, which was found to improve the network convergence. The filters of convolutional layers are with a small receptive field 3×3, and all the convolution stride is fixed to one. The number of feature maps is increased from 50 (layer-1) to 400 (layer-12) gradually. To further increase the depth of the network so as to improve the classification capability, the spatial pooling is implemented after every three convolutional layers instead of two in [40], which is carried out by max-pooling (over a 2×2 window with stride 2) to halve the size of feature map. After the stack of 12 convolutional layers and 4 max-pool layers, the feature maps are flattened and concatenated into a vector with dimensionality 1600. Two fully-connected

layers (with 900 and 200 hidden units respectively) then follows. At last, the softmax layer is used to perform the 7357-way classification, including 7356 character classes and one non-character class. The extra non-character class unit is to explicitly reject non-characters, which are generated frequently in text line recognition [71]. Wang et al. [32] have found that it is better to directly add an extra negative class other than using the cascading CNN.

5.2. Over-segmentation

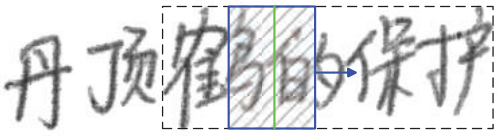


Figure 5: Sliding window based over-segmentation.

Over-segmentation is to separate a text line image into primitive segments, each being a character or a part of a character, such that characters can be formed by concatenating consecutive primitive segments. Connected component (CC) analysis has been commonly used for over-segmentation in Chinese text recognition, but the splitting of touched Chinese character is still critical to the performance of text recognition. The conventional splitting method based on profile shape analysis [33] has been applied successfully in many works [3, 12, 21], but it fails in dealing with complex touching situations, as is shown in Fig. 6(a).

For improving the character separation rate, we adopt a two-stage CNN based over-segmentation method in this work:

(1) The text line image is initially over-segmented into primitive segments using the visibility-based foreground analysis method proposed in [42]. The position between two adjacent primitive segments is a candidate segmentation point.

(2) A binary output CNN is used to classify sliding windows on CCs generated in step 1 for detecting more candidate segmentation points. Detected segmentation points close to each other are suppressed heuristically.

Our previous work on neural network based over-segmentation has demonstrated effective in scene text recognition [72]. In this work, we improve the algorithm in two aspects. Firstly, the visibility-based foreground analysis for over-segmentation [42] before sliding window detection is complementary to the sliding window method, and can speed up the subsequent operation. Secondly, we use CNN as the classifier rather than a traditional neural network, for higher detection rate of segmentation points. The step 2 is elaborated in the following.

On the image of a CC, a fixed-width window slides from left to right with a stride of 0.1 times the CC height, as depicted in Fig. 5. The window image is classified by a CNN to judge whether the center column is a segmentation point or not. The window has the same height as the CC, and the width of 0.8 times the CC height. We observed experimentally that the segmentation and recognition performance is insensitive to the stride coefficient ranged from 0.04 to 0.1 and the window width ranged from 0.6 to 1 times the CC height.

When training a CNN for segment point classification, a complex structure, such as the one in [40], is prone to overfitting. Hence, we built a simple 4-layer network for binary classification, as shown in Table 2. This network also uses extended directMaps mentioned above as input. The CNN is trained using window image samples with the center positions labeled as segmentation point (positive) or not (negative). On a CC image, after segmentation point detection by sliding window classification, we merge adjacent candidate segmentation points which are close to each other, i.e., horizontal distance less than a threshold, and retain the one with the smallest vertical projection. The threshold is empirically set as the stroke width, which is estimated from the contour length and foreground pixel number in the text line image. Fig. 6 shows some examples of over-segmentation, where we can see that the method of [42] is slightly better at recall rate than that of [33], and our proposed method can separate touching characters better than both the methods of [33] and [42].

5.3. Geometric context models

Geometric context models have been successfully used in character string recognition [3, 35], and transcript mapping [57], where they play an important role to exclude

Table 2: CNN configuration for over-segmentation.

Type	Configurations
input	$9 \times 32 \times 32$ extended directMaps
Convolution	#maps: 50, k: 3×3, s:1, p:1, dropout: 0.0
MaxPooling	Window: 2×2, s: 2
Convolution	#maps: 100, k: 3×3, s:1, p:1, dropout: 0.1
MaxPooling	Window: 2×2, s: 2
Full connection	#units: 200
Softmax	#units: 2

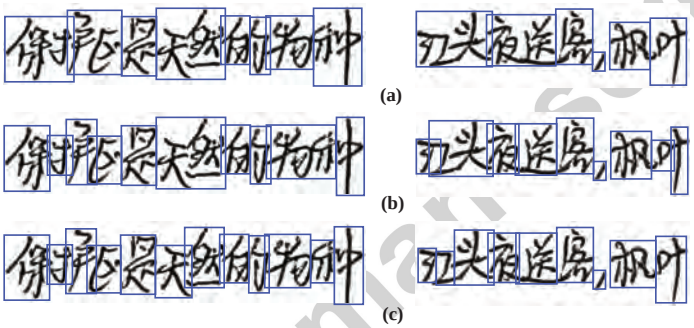


Figure 6: Examples of over-segmentation. (a) traditional method [33]. (b) Xu et al. [42]. (c) Our method.

non-characters and further improve the system performance. In this study, we adopt the framework of geometric context model presented in [57], where geometric context is divided into four statistical models (unary and binary class-dependent, unary and binary class-independent), abbreviated as **ucg**, **bcg**, **uig**, **big**, respectively.

The class-dependent geometric model can be seen as a complement to the character classifier since the candidate patterns retain their original outlines without normalization designed for character classification, which may exclude some useful context

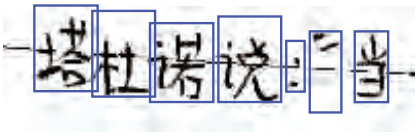


Figure 7: The process of polynomial curve fitting.

information related to writing styles. Following [57], we reduce the number of character geometry classes to six super-classes. The **uig** model is used to measure whether a candidate pattern is a valid character or not, while the **big** model is used to measure whether an over-segmentation gap is a valid between-character gap or not. They are both two-class (binary classification) models.

For modeling the four geometric models, we used to extract geometric features firstly, and then use quadratic discriminant function (**ucg**, **bcg**) or support vector machine (**uig**, **big**) for classification, and finally transform the output scores to probabilities by confidence transformation. In this work, we utilize CNN to perform feature extraction and classification in a unified framework, then directly use the output of a specific unit as the final score. Instead of simply resizing the character patterns as the input, we acquire the center curve of the text line by polynomial fitting which is shown in Fig. 7, as it is necessary to keep the writing styles of text lines for geometric context models. The degree of polynomial is set to be 0.075 times the connected component number. After that, the top and bottom boundaries of each CC are adjusted according to the center curve and the character height. In this case, we use the same CNN architecture as the one in [40] except for different units for output layers. In order to maintain the writing styles, we only use the original CC image as input without directMaps.

6. Experimental results

We evaluated the performance of our handwritten Chinese text recognition system on two databases: a large database of offline Chinese handwriting called CASIA-HWDB [54], and a small dataset from the ICDAR 2013 Chinese Handwriting Recognition Competition [36], abbreviated as ICDAR-2013. The system was implemented on a desktop computer of Intel Core i7-4790 3.60 GHz CPU, programming with C++ in Microsoft Visual Studio 2008. While for training NNLMs and CNN shape models, we also used NVIDIA Titan X GPUs for acceleration.

6.1. Database and baseline experimental setup

The CASIA-HWDB database contains both isolated characters and unconstrained handwritten texts, which is divided into a training set of 816 writers data and a test

set of 204 writers data. The training set contains 3,118,447 isolated character samples of 7,356 classes and 4,076 pages of handwritten pages (including 1,080,017 character samples). We tested our system on the test set containing 1,015 pages. The ICDAR-2013 dataset was used as the test set at the ICDAR 2013 competition. It contains 300 test pages, which were written by 60 writers who did not contribute to the released CASIA-HWDB database.

In the first round of experiments, to compare our results with the best ones with similar setup reported in [3, 12] fairly, we used the same character classifier and geometric context models trained on the CASIA-HWDB training set, the same over-segmentation technique and the same text corpus for training LMs, as detailed in [3]. While in the second round of experiments, we replace the traditional character classifier, over-segmentation and geometric context models with CNN based models. In the third round, we further switch to a large text corpus for training LMs.

The character classifier was trained on 4,198,494 isolated character images of 7,356 classes from both isolated characters and unconstrained texts. From a character image, 512-dimensional gradient direction features are extracted from gray-scale image using the method of normalization cooperated gradient feature (NCGF) [73]. The 512D feature vector is reduced to 160D by Fisher linear discriminant analysis (FLDA), and then input into the Modified Quadratic Discriminant Function (MQDF) [74] classifier for assigning candidate classes and confidence scores. We used 4/5 samples of the training set for training the classifiers, and the remaining 1/5 samples for estimating the confidence parameter (for transforming classifier output scores to posterior probabilities, details can be found in [3]).

To build the geometric context models [57], we extracted geometric features from 41,781 text lines of training text pages for estimating the parameters of the corresponding four models (classifiers on geometric features) (**ucg**, **uig**, **bcg**, and **big**).

The generic language models were trained on a text corpus containing about 50 million characters, which is the same as that in [3]. For comparison with the results in [12], we also trained language models on the same large corpus, which contains the above general corpus and the corpus from Sogou Labs, containing approximately 1.6 billion characters. In addition, we collected a development set containing 3.8 million

characters from the People's Daily corpus [75] and ToRCH2009 corpus [76], for validating the trained language models. In the baseline setup, the maximum number of concatenated segments, candidate number of character classification and beam width in the refined beam search algorithm are set as 4, 20 and 10, respectively.

We report recognition performance in terms of two character-level metrics following [77]: Correct Rate (CR) and Accurate Rate (AR):

$$\begin{aligned} CR &= (N_t - D_e - S_s)/N_t, \\ AR &= (N_t - D_e - S_s - I_e)/N_t, \end{aligned} \tag{15}$$

where N_t is the total number of characters in the transcript of test documents. The numbers of substitution errors (S_e), deletion errors (D_e) and insertion errors (I_e) are calculated by aligning the recognition result string with the transcript by dynamic programming. In addition to the AR and CR, we also measured the PPL of language models on the development set.

Since our experiments involve many context models, we give a list of the models in Table 3.

Table 3: List of context models used in handwritten Chinese text recognition.

Abbreviation	Referred model
cls	character classifier (MQDF)
g	union of all geometric models
cbi	character bigram language model
cti	character trigram language model
cfour	character 4-gram language model
cfive	character 5-gram language model
rnn	character recurrent neural network language model
iwc	interpolating word and class bigram

6.2. Comparison of language models

The first round of experiments is to compare the recognition performance using language models trained on the general corpus and traditional over-segmentation in the system. For comparison, we first give the baseline results of our system using BLMs

of different orders on the CASIA-HWDB test set. Then, we evaluate and compare the performance of FNNLMs and RNNLMs of various structures. Last, we report the recognition performance on the ICDAR-2013 competition set with NNLMs.

6.2.1. Baseline performance

The baseline recognition performance is obtained using BLMs trained on the general corpus. To be consistent to the previous works [3, 12], we set the maximum number of concatenated primitive segments as 4, the number of top candidate classes in classification as 20, and the beam width in the refined beam search algorithm as 10. The bigram, trigram, 4-gram, and 5-gram BLMs were trained with the SRI Language Model (SRILM) toolkit (1.7.1) [78] with the default smoothing technique (Katz smoothing) and entropy-based pruning. The thresholds for pruning the character bigram, trigram, 4-gram and 5-gram are set empirically as 5×10^{-8}, 10^{-7}, 10^{-7} and 10^{-7}, respectively. The parameters for the bigram and trigram are the same as those in [3].

The recognition results using different combinations of context models on the CASIA-HWDB test set are shown in Table 4. Naturally, our results using language models **cbi** and **cti** are almost identical to those reported in [3] based on same settings of parameters. We achieved faster recognition than [3] due to the difference of computer hardware and some details of implementation. The results of BLMs of different orders show that while the improvement from **cbi** to **cti** is remarkable, the improvement from **cti** to higher order LMs **cfour** and **cfive** is only marginal. This can be attributed to the data sparseness problem, which affects higher order LMs more evidently and cancels off the benefit of higher order LMs.

Table 4: Recognition results using BLMs. "Time" denotes the recognition time on all the test pages.

Combination	AR (%)	CR (%)	Time (h)	PPL
cls+cbi+g [3]	89.56	90.27	11.33	-
cls+cti+g [3]	90.20	90.80	11.54	-
cls+cbi+g	89.57	90.28	6.60	144.81
cls+cti+g	90.21	90.81	6.68	82.97
cls+cfour+g	90.23	90.82	6.72	73.72
cls+cfive+g	90.23	90.82	6.84	73.09

6.2.2. Effects of FNNLMs

We trained the FNNLMs of various structures with the CSLM toolkit (v3) [79], which provides full support for short-list and GPU implementation. We also used Intel Math Kernel Library (MKL) to speed up the matrix operations of neural network in testing. Each structure was trained multiple times with different initializations, and the model with the lowest PPL on the development set was chosen as the final one.

We evaluated three structures of FNNLMs with projection size of 320, as shown in Table 5. The networks were all trained with batch size of 128 examples, weight decay coefficient 10^{-7}, and 20 iterations. The structures FNNLM-1 and FNNLM-2 have two hidden layers, while FNNLM-3 has only one hidden layer. The short-list of FNNLM-1 covers all the characters in the training corpus, while FNNLM-2 and FNNLM-3 use a smaller short-list. The learning rate $lrate$ was empirically set initial values as in Table 5, and decreased gradually during training by the following equation:

$$lrate = lrate_0/(1 + \lambda n) \tag{16}$$

where $lrate_0$ is the initial learning rate, n denotes the number of totally seen samples, λ stands for the decay parameter and is 5×10^{-8} in this paper. As discussed in Section 4.3, we also constructed HLMs by linearly interpolating FNNLMs with standard BLMs. The weights of interpolation were computed by the compute-best-mix-tool from the SRILM toolkit, minimizing the perplexity on the development set. Corresponding to the three structures of FNNLMs, we have three HLMs denoted as HFLM-1, HFLM-2, and HFLM-3, respectively.

Table 5: Three structures of FNNLMs.

Structure	Hidden layer size	Short-list length	Initial learning rate
FNNLM-1	1024×512	8330	0.06
FNNLM-2	1024×512	1023	0.06
FNNLM-3	512	1023	0.10

The recognition results using different combinations of context models are shown in Table 6. Since it is quite time-consuming to test NNLM-1 and NNLM-2 of various

orders, we only evaluated these two structures with 5-gram, which usually outperforms 4-gram and 3-gram. Comparing the results of FNNLMs with those of BLMs in Table 4, we can see that though FNNLMs yields lower PPL than the BLMs of same order, their benefit on the recognition performance is not evident. The hybrid model, i.e., interpolation of FNNLM and BLM, can further reduce the PPL and evidently improve the text recognition accuracies. Specifically, the 5-gram HFLM-1 improves the AR and CR to 90.69% and 91.24%, respectively, compared to 90.23% AR and 90.82% CR of the 5-gram BLM. The FNNLM-2, with a short-list of size 1023, has much lower complexity than the FNNLM-1, and yields slightly lower recognition performance, which is comparable to that of 5-gram BLM. The corresponding hybrid model, 5-gram HFLM-2, yields lower performance than the 5-gram HFLM-1, but its time is greatly reduced by 87.09% compared to HFLM-1, and the performance is superior to that of BLM and FNNLM-2.

The FNNLM-3, with only one hidden layer and also short-list, has much lower complexity than the FNNLM-1 and FNNLM-2. Consequently, its performance also degenerates in terms of both PPL and recognition accuracy. The AR and CR of FNNLM-3 are even lower than those of the BLM of same order. However, when combining with BLMs, the resulting hybrid model HFLM-3 performs much better than the FNNLM-3. The recognition performance of 5-gram HFLM-3 is even comparable to that of the HFLM-2 of same order, which has two hidden layers and consumes much longer time than the HFLM-3. These results confirm that a hybrid LM by combining a simple-structure FNNLM with a BLM is a good choice to balance the computational complexity and the recognition performance.

6.2.3. Effects of RNNLMs

We trained the RNNLMs with the RNNLM Toolkit (0.4b) [65], which provides support for output factorization, RNNME training and efficient computation, although without parallel computation. In order to make fair comparison with FNNLMs, we also modified the toolkit to allow RNNLMs training with short-list. Since the RNNLM Toolkit does not support parallel computation, we only trained the RNNLMs with short-list or output factorization, while the training of RNNLMs without reduction is

Table 6: Recognition results using FNNLMs and HLMs.

Language model	Combination	AR (%)	CR (%)	Time (h)	PPL
FNNLM-1	cls+cfive+g	90.29	90.88	129.30	68.60
HFLM-1	cls+cfive+g	90.69	91.24	140.32	59.44
FNNLM-2	cls+cfive+g	90.21	90.82	17.78	71.64
HFLM-2	cls+cfive+g	90.51	91.09	18.11	63.03
FNNLM-3	cls+cbi+g	89.59	90.30	9.06	141.39
	cls+cti+g	90.00	90.64	9.45	87.18
	cls+cfour+g	90.04	90.66	10.04	79.63
	cls+cfive+g	90.12	90.75	10.52	76.75
HFLM-3	cls+cbi+g	89.62	90.32	8.92	140.38
	cls+cti+g	90.33	90.92	9.73	66.83
	cls+cfour+g	90.39	90.97	10.21	66.83
	cls+cfive+g	90.49	91.07	10.81	64.66

intractable for large vocabulary.

To investigate the effect of hidden layer number on RNNLMs, we trained two structures of RNNLMs with short-list size 1023, SRNNLM-1 and SRNNLM-2, which have hidden layer size of 300 and 600, respectively. The other parameters are listed in Table 7. The learning rate is constantly halved once the log-likelihood improvement rate on the development set is lower than the minimum improvement ratio.

Table 7: Training parameters of RNNLMs.

BPTT step	BPTT block	Initial learning rate	Weight decay	Minimum improvement ratio
6	10	0.1	10^{-7}	1.003

We compare the performance of RNNLMs and FNNLMs both with short-list, and also evaluate the hybrid models by combining RNNLM with BLM. Since RNNLMs are no longer limited to N-gram models, it is not worthy to interpolate with BLMs of low order. Hence, we only combined RNNLMs with 5-gram BLMs. The two hybrid models corresponding to SRNNLM-1 and SRNNLM-2 are referred to as HSRLM-1 and HSRLM-2, respectively. The recognition results are shown in Table 8.

Table 8: Effects of short-list RNNLMs and HLMs.

Language type	Combination	AR (%)	CR (%)	Time (h)	PPL
SRNNLM-1	cls+rnn+g	89.88	90.57	14.30	81.49
HSRLM-1	cls+rnn+g	90.43	91.02	14.42	61.44
SRNNLM-2	cls+rnn+g	90.30	90.94	28.74	65.19
HSRLM-2	cls+rnn+g	90.61	91.20	28.99	55.86

We mainly compare SRNNLM-1 and SRNNLM-2 with 5-gram FNNLM-2 and FNNLM-3, which do not differ largely in complexity. From the characteristics of the networks, we know that the order of parameter size of the four different models is SRNNLM-2(6M)>FNNLM-2(5M)>FNNLM-3(4M)>SRNNLM-1(3M). Compare the results of SRNNLMs in Table 8 and FNNLMs in Table 6, we can see that the SRNNLM-1 yields the lowest performance among these four models. However, once interpolated with the 5-gram BLM, the HSRLM-1 has lower PPL than the 5-gram HFLM-2 and HFLM-3, and perform comparably with them in recognition. This indicates that SRNNLMs provide better complementarity to BLMs than FNNLMs, such that simple-structure SRNNLMs generates competitive HLMs. Although the SRNNLM-2 is slightly more complex than the FNNLM-2, it performs evidently better than the 5-gram FNNLM-2. When interpolating with BLMs, the corresponding HSRLM-2 also outperforms the HFLM-2 in both PPL and recognition accuracies. In fact, the HSRLM-2 even performs comparably with the HFLM-1 in Table 6, where the 5-gram FNNLM-1 has roughly 9M parameters and is much more time consuming than the HSRLM-2. Overall, these results demonstrate the superior performance of RNNLMs over FNNLMs of similar parameter complexity. As for the time complexity, the RNNLM is a little slower in our recognition system because of the frequent exchange of memory for hidden layers.

We also compared the performance of short-list RNNLM with output factorized RNNLM (FRNNLM) and RNNME. As the vocabulary size V from the general corpus is 8330, we set the word class number as 100, which is close to the suggested number $\sqrt{|V|}$ in [65], and . The RNNME model in our experiments has hidden layer size

of 300 and uses 4-gram features[3] with hash array size of 100M. The recognition results of factorized RNNLMs and RNNME models as well as their hybrid models are shown in Table 9, where FRNNLM-1 and FRNNLM-2 denote the factorized RNNLMs with hidden layer size of 300 and 600, respectively, HFRLM-1 and HFRLM-2 are the corresponding HLMs linearly interpolated with 5-gram BLMs, HRMELM denotes the RNNME based HLM interpolated with a 5-gram BLM.

Table 9: Recognition results of factorized RNNLMs and RNNME models.

Language type	Combination	AR (%)	CR (%)	Time (h)	PPL
FRNNLM-1	cls+rnn+g	89.57	90.26	15.95	87.67
HFRLM-1	cls+rnn+g	90.47	91.02	15.99	60.22
FRNNLM-2	cls+rnn+g	90.03	90.70	31.36	70.95
HFRLM-2	cls+rnn+g	90.61	91.16	31.61	55.30
RNNME	cls+rnn+g	90.89	91.38	16.44	56.50
HRMELM	cls+rnn+g	**91.04**	**91.52**	16.48	52.92
BLM	cls+iwc+g+cca* [3]	90.75	91.39	18.78	-

cca: candidate character augmentation

First, compared to the results of short-list RNNLMs in Table 8, we can see that the FRNNLM-1 performs even worse than the SRNNLM-1 with the same hidden layer size of 300, since the network size is too small to capture the context for full vocabulary. However, the hybrid model HFRLM-1 outperforms the HSRLM-1, as the full vocabulary (though factorized) output layer can offer lager potential for correct recognition. When increasing the hidden layer size to 600, the FRNNLM-2 again performs worse than the SRNNLM-2, while the hybrid models HFRLM-2 and HSRLM-2 perform comparably. In terms of computation efficiency, the short-list method turns out to be more efficient than output factorization.

Next, we compare the performance of factorized RNNLMs and RNNME in Table 9. Since the factorized RNNLM with hidden layer size of 600 is still not sufficient for full vocabulary on the general corpus, we introduced the RNNME into our system. It can be seen that the RNNME alone can greatly improve the recognition performance, even

[3]According to [65], maximum entropy models with up to 4-gram features perform sufficiently.

outperform the hybrid model HFRLM-2, although its PPL is slightly higher than that of the HFLM-2. In fact, the RNNME alone has yielded performance (in terms of AR) superior to the state-of-the-art result reported in [3], which was obtained using a candidate character augmentation (CCA) technique to promote the probability of including the correct class of candidate character patterns on the segmentation-recognition lattice, while we did not use CCA in this work. By interpolating 5-gram BLM, the hybrid model HRMELM yields the best results of 91.04% AR and 91.52% CR.

6.2.4. Performance on ICDAR-2013 dataset

Since the test set of ICDAR 2013 Chinese handwriting recognition competition is now widely taken for benchmarking, we also report results on this dataset. We present recognition results with three types of language models: 5-gram BLM, RNNME, and HRMELM. The language models and the text recognition settings are all the same as those for recognition on the CASIA-HWDB test set. The recognition results are shown in Table 10.

Table 10: Recognition results on ICDAR-2013 dataset.

Language model type	Combination	AR (%)	CR (%)	Time (h)	PPL
BLM	cls+iwc+g+cca [3]	89.28	90.22	-	-
BLM	cls+cfive+g	89.03	89.91	2.44	73.09
RNNME	cls+rnn+g	89.69	90.41	5.86	56.50
HRMELM	cls+rnn+g	**89.86**	**90.58**	5.84	52.92

Table 10 also gives the results of interpolated word class (**iwc**) bigram with CCA. Due to the effect of CCA, the iwc bigram even outperforms the character-based 5-gram BLM. The comparison of 5-gram BLM, RNNME and HRMELM show similar tendency as in Table 9: the RNNME outperforms the 5-gram BLM, and the HRMELM yields the best performance. Compared to the state-of-the-art result of the method in [3], the error rate is reduced by 5.41% relative with only the help of character level language models.

6.3. Effects of CNN shape models

In the second round of experiments, we replace the traditional character classifier, over-segmentation, and geometric context models with CNN shape models, which are all trained with Caffe [80]. We first introduce the training details of the three models. Then we evaluate the performance of these models and compare with traditional ones. For both the character classifier and geometric models, we directly use the softmax output as the corresponding score without confidence transformation.

6.3.1. CNN character classifier

The CNN character classifier was initialized using Xavier initialization [81]. The training is carried out by minimizing the multi-class negative log-likelihood loss using mini-batch gradient descent with momentum. The batch size is set to be 1024, while the momentum is 0.9. The learning rate is initially set to 0.01, and then decreased by $\times 0.5$ when the cost or accuracy on the training data stops improving. The training can be finished after about 90 epochs.

Our experiments showed that although training data augmentation does not improve the character recognition accuracy, it can improve the string recognition performance. We adopted the augmentation techniques introduced in [82], where geometric transform, local resizing and elastic distortion are used. We expanded the training set by two times of the original samples, i.e., we totally had 12,595,460 character samples. On the other hand, we also generated 5,160,425 non-character samples from the training text line samples. Although there exits a severe class imbalance problem, we found no obvious performance deterioration in the system performance. Hence, we did not utilize any technique to deal with this problem. The CNN character classifier achieved 92.17% accuracy on the character samples segmented from the test text set of CASIA-HWDB. Compared with the accuracy 83.78% in [3], the CNN classifier is obviously much stronger than MQDF.

6.3.2. CNN based over-segmentation

For over-segmentation, we trained the sliding window classifier (CNN) with the training data of CASIA-HWDB database. Since all the text lines in the database have

been segmented and annotated at character level, it is convenient to get the ground-truths of segmentation points. For generating training samples, we first slide the window on CCs in the text lines. When the distance between the center position of the window and the boundary of a character is smaller than 0.1 times the CC height or larger than 0.12 times the CC height, the window is regarded as a positive or negative sample, respectively. Otherwise, the window is regarded ambiguous and not used for training.

The initialization and training procedure are similar to those of the CNN character classifier. There are usually much more negative samples (1,870,534) than positive samples (123,862). To overcome the problem of sample class imbalance and improve the recall as much as possible, we decreased the negative sample loss by $\times 0.005$. The training can be finished after about 100 epochs.

For evaluating over-segmentation on text lines, we measured the precision and recall rate of segmentation point detection. When the window classifier outputs positive label, if the horizontal distance between the window center and a character boundary is less than 2 times the stroke width of the text line, the window center is regarded as a true positive of segmentation point, otherwise is a false positive. The segmentation precision and recall rates on the CASIA-HWDB test set are shown in Table 11. We can see that the method of [42] can achieve higher recall than that of [33] at a little loss of precision. When using sliding window classification, the recall rate is further improved compared to both the methods of [33] and [42]. Our CNN based method, by combining the method in [42] with siding window classification, can achieve the highest recall rate, which offers higher potential of correct character segmentation and recognition. In the recognition system, as the CNN based over-segmentation algorithm generates more CCs, the maximum number of concatenated segments is set to be 7 instead of 4.

6.3.3. CNN based geometric context models

Since the text lines in the database HWDB2.0-2.2 were annotated at character level, it is convenient to get the ground-truths for the four types of geometry samples (**ucg**, **uig**, **bcg**, **big**). We have got 1,081,153 (**ucg**), 7,498,977 (**uig**), 1,221,326 (**bcg**) and 1,331,428 (**big**) samples, respectively. The initialization and training procedure are

Table 11: Over-segmentation results on CASIA-HWDB test set.

Model type	Precision (%)	Recall (%)
[33]	74.32	98.23
[42]	68.07	99.22
Only sliding window	63.75	99.39
Our method	64.23	99.58

similar to the CNN character classifier as well.

6.3.4. Evaluation of CNN shape models

We integrated the three CNN-based models above into the recognition system to validate the improvement of performance. Based on the former comparison of different LMs, we only used one best NNLM, the HRMELM. The recognition results on the CASIA-HWDB database and the ICDAR-2013 dataset are listed in Table 12.

Table 12: Recognition results using different type of models on two datasets.

Shape model	LM	Combination	CASIA-HWDB			ICDAR-2013		
			AR (%)	CR (%)	Time (h)	AR (%)	CR (%)	Time (h)
traditional	BLM	cls+iwc+g+cca [3]	90.75	91.39	18.78	89.28	90.22	-
	BLM	cls+cti+g+lma* [12]	91.73	92.37	10.17	-	-	-
	BLM	cls+cfive+g	90.23	90.82	6.84	89.03	89.91	2.44
	HRMELM	cls+rnn+g	91.04	91.52	16.48	89.86	90.58	5.84
CNN	BLM	cls+cfive+g	95.05	95.15	8.67	94.51	94.64	2.96
	HRMELM	cls+rnn+g	**95.55**	**95.63**	16.83	**95.04**	**95.15**	6.68

lma: language model adaptation

From Table 12, the comparison between the traditional models and CNN based models in this work shows the superiority of the CNN. The Character Error Rate (CER), which equals $1 - AR$, is reduced by almost 50% compared to the traditional models on both two datasets. Furthermore, text recognition based on CNN models consumes only a little more computation time than the traditional ones, because of the highly efficient implementation of our algorithm on GPU. The comparison between

different LMs again validates the superiority of the HRMELM. Combined with CNN shape models, the HRMELM yields the best performance on both two datasets, i.e., 95.55% AR and 95.63% CR on the CASIA-HWDB test set, 95.04% AR and 95.15% CR on the ICDAR-2013 dataset. These were resulted without using CCA (candidate character augmentation) or LMA (language model adaptation, based on a large corpus classified into different domains).

For references, the ICDAR-2013 competition paper [36] reported best results of 89.28% AR and 90.22% CR using the method of [3]. The work [13] implemented the LSTM-RNN framework (initially introduced in [83]) for Chinese handwritten text recognition and reported promising recognition performance on the ICDAR-2013 dataset: 89.40% AR. This is inferior to the performance of the proposed method using HRMELM with either traditional models or CNN shape models. A recent work [32], which adopts a similar framework to ours, achieves 95.21% AR and 96.28% CR on the CASIA-HWDB test set, 94.02% AR and 95.53% CR on the ICDAR-2013 dataset. Our method has much lower CER, which is supposed to be a more accurate metric, although it is a little worse than [32] in CR. Moreover, it should be mentioned that both [13] and [32] removed some special tokens when tested on the ICDAR-2013 dataset.

6.4. Results with LMs on large corpus

To better model linguistic contexts, we extended our experiments using a large corpus containing 1.6 billion characters, which was used in a previous work of language model adaptation [33]. On the large corpus, we trained a 5-gram BLM with the same Katz smoothing, and also set the threshold of pruning as 10^{-7}. Since it is too time consuming to train NNLMs on the large corpus, we simply used the NNLMs trained on the general corpus containing 50 millions of characters, and combined them with BLMs trained on the large corpus. Particularly, we used the RNNME model trained on the general corpus and combined it with the 5-gram BLM trained on the large corpus to give a hybrid model HRMELM. The recognition results on two datasets are shown in Table 13.

From Table 13, we have three observations on the results on CASIA-HWDB. First, unlike the performance of higher order LMs trained on the smaller general corpus, the

Table 13: Recognition results on two datasets using LMs on large corpus.

LM type	Shape model	Combination	CASIA-HWDB				ICDAR-2013			
			AR (%)	CR (%)	Time (h)	PPL	AR (%)	CR (%)	Time (h)	PPL
BLM	traditional	cls+cti+g+lma[12]	91.73	92.37	10.17	-	-	-	-	-
	traditional	cls+cti+g[12]	90.66	91.28	9.83	-	-	-	-	-
	traditional	cls+cfive+g	90.79	91.41	6.88	65.21	91.48	92.17	2.44	65.21
	CNN	cls+cfive+g	95.36	95.46	8.91	65.21	96.18	96.31	2.93	65.21
HRMELM	traditional	cls+rnn+g	91.64	92.11	16.57	46.25	91.59	92.23	5.93	46.25
	CNN	cls+rnn+g	**95.88**	**95.95**	16.83	46.25	**96.20**	**96.32**	5.93	46.25

5-gram BLM obviously outperforms the 3-gram **cti** when trained on large corpus, since the large corpus alleviates the data sparseness problem. Second, although the performance of the 5-gram BLM is improved by the large corpus, the RNNME still benefits the performance significantly in the HRMELM: it brings 9.23% error rate reduction compared to the 5-gram BLM. Third, CNN shape models again improve the system performance in the context of large corpus, because they not only provide larger potential of containing correct candidate character patterns, but also offer stronger classification capabilities. Compared to the previous state-of-the-art baseline [12] using **lma** on large corpus, our method using HRMELM and CNN shape models improves the AR by 4.15% absolutely.

It is noteworthy in Table 13 that when using the large corpus for training LMs, the HRMELM shows no obvious superiority to the BLM on the ICDAR-2013 dataset. We found that the transcripts of the ICDAR-2013 dataset are mostly included in the corpus from Sogou Labs, thus, the BLM can fit the test data very well and yields high recognition accuracies. To further investigate into this problem, we deleted the sentences which appear in the ICDAR-2013 corpus from the large corpus. However, as the topics of this corpus are very typical and concentrated, on the ICDAR-2013 dataset, we can achieve 96.16% AR and 96.29% CR with the 5-gram back-off LM trained on the processed corpus as well. This alerts researchers to pay attention to the overfitting of language model to the transcripts of test text images. On the other hand, neural

network LMs generalize well to unseen texts.

6.5. Performance analysis

The experimental results show that there is still a gap between the accuracy and perfect recognition despite the improvements of NNLMs and CNN based models. Thus, we analyze the upper bound of performance of our recognition system in the following, and then show some real examples of text line recognition.

6.5.1. Upper bound of performance

For a quantitative measure of upper bound of recognition performance, we consider the Lattice Error Rate (LER) to evaluate the quality of the segmentation-recognition candidate lattice. The LER is a lower bound of CER. It is defined as the total number of lattice errors divided by the total number of characters in the transcript. The calculation of LER is specified as follows.

For a text line (character string) S with transcript $\mathbf{y}_{1:N} = y_1...y_N$, we denote the lattice of S by **G**. The lattice errors are evaluated by the distance between $\mathbf{y}_{1:N}$ and **G**, which is defined as the minimum edit distance between $\mathbf{y}_{1:N}$ and any label sequence Y in **G**:

$$Dist(\mathbf{y}_{1:N}, \mathrm{G}) = \min_{Y \in \mathbf{G}} Dist(\mathbf{y}_{1:N}, Y). \tag{17}$$

Let $\mathbf{y}_{1:i}$, $i \leq N$, be a partial label sequence. Let $j \in \{0, ..., T\}$ be the candidate segmentation points of text line, where 0 is the start and T is the end, and $\mathbf{G}_{0:j}$ be the partial lattice between segmentation points 0 and j. Then the distance $D(i, j)$ between $\mathbf{y}_{1:i}$ and $\mathbf{G}_{0:j}$ can be deduced recursively by the following dynamic programming procedure:

(1) Initialization

$$\begin{aligned} &D(0,0) = 0, \\ &D(i,0) = i, \text{ for } 1 \leq i \leq N, \\ &D(0,j) = \min_{k:(k,j)\in\mathbf{G}} D(0,k) + 1, \text{ for } 1 \leq i \leq T \end{aligned} \tag{18}$$

(2) Recursion. For $1 \leq i \leq N, 1 \leq j \leq T$,

$$D(i,j) = \min \begin{cases} \min\limits_{k, Y_{(k,j)}:(k,j)\in \mathbf{G}} D(i-1,k) + 1 - \delta(y_i, Y_{(k,j)}), \\ D(i-1,j) + 1, \\ \min\limits_{k:(k,j)\in \mathbf{G}} D(i,k) + 1, \end{cases} \tag{19}$$

(3) Termination

$$Dist(\mathbf{y}_{1:N}, \mathrm{G}) = D(N,T), \tag{20}$$

where (k,j) with $k < j$ denotes a candidate character pattern between candidate segmentation points k and j, and $Y_{(k,j)}$ denotes a candidate class of (k,j).

We can see that the LER is affected by over-segmentation and candidate character classification, which generates character segmentation and assigns character classes. Combined with the traditional and CNN shape models respectively, we compare the LERs of these two models. We show the Lattice Accuracy Rate (LAR) as $1 -$ LER in Table 14. It is shown that the CNN shape models, specifically the character classifier and over-segmentation algorithm, significantly improve the LAR (reduces the LER) on both two datasets. According to [3], the baseline MQDF classifier gives top-20 cumulative accuracy 98.24% on the characters in the test text lines of CASIA-HWDB, while our CNN classifier achieves 99.75% top-20 cumulative accuracy. However, the gap between the actual AR (usually less than 96% in our experiments) and the LAR implies that there is still room for improvement in exploiting contexts on the segmentation-recognition candidate lattice.

Table 14: LARs on the two datasets.

Shape model	CASIA-HWDB	ICDAR-2013
	LAR(%)	LAR(%)
traditional	96.65	96.16
CNN	99.20	99.27

6.5.2. *Recognition examples*

We show some examples of text line recognition in Fig. 8, which reveal several factors causing recognition errors. We consider two typical settings: the 5-gram BLM combined with traditional models, the best language model HRMELM combined with CNN shape models. The four examples show the effects of both language models and context evaluation models. The recognition error in Fig. 8(a) was also shown in [12], and was not corrected by language model adaptation. It is corrected by the HRMELM which captures long-span context. In (b) the error is corrected by the CNN based character classifier and geometric models for better modeling the contexts. In (c), the error is corrected by CNN based over-segmentation, while the tradition over-segmentation method could not separate the two touched characters. In (d), the error is irreducible due to the inaccuracy of candidate segmentation-recognition path evaluation.

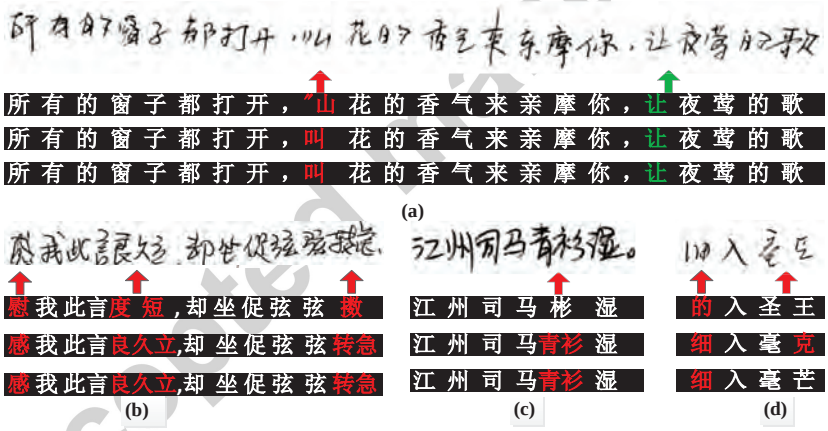


Figure 8: Recognition of four text lines. For each example, the first row is the text line image, second row is the result using 5-gram BLM and traditional models, third row is the result using HRMELM and CNN shape models, fourth row is the transcript (ground-truth).

7. Conclusion

In this paper, we evaluated the effects of two types of character-level NNLMs, namely, FNNLMs and RNNLMs, with the aim of improving Chinese handwritten text recognition. Both FNNLMs and RNNLMs are also combined with BLMs to construct

HLMs. We evaluated in a text line recognition system with the same character over-segmentation and classification techniques as in a state-of-the-art system, and compared various LMs trained on a small text corpus as used before. Experimental results on the Chinese handwriting database CASIA-HWDB show that while pure NNLMs do not improve the recognition performance substantially, the hybrid LMs by combining NNLMs and BLMs lead to significant improvements. RNNLMs outperform FNNLMs because they can model long-distance contexts. The hybrid model HRMELM (combining the RNNME and BLM) yields the best performance. Replacing traditional character classifier, over-segmentation and geometric context model with CNN based models and training LMs with a large corpus, we achieved new benchmarks on both the CASIA-HWDB database and the ICDAR-2013 competition dataset.

The analysis of recognition performance upper bound (LAR) and examples of recognition errors show that there is still large room for improvements, mainly lying in candidate segmentation-recognition path evaluation exploiting contexts. In our future work, we will consider the more powerful LSTM-RNN language model, which is even more computationally demanding than the RNNLMs. We will also try word level LMs, as words are more semantically meaningful than pure characters, though word level LMs are hard to manage in Chinese documents.

Acknowledgements

We would like to thank Zhuo Chen and Xin He for the help of implementing CNN based over-segmentation, and Xiang-Dong Zhou for sharing the idea of lattice error rate. This work has been supported by the National Natural Science Foundation of China (NSFC) grants 61305005, 61273269, 61573355, and 61411136002.

References

[1] R.-W. Dai, C.-L. Liu, B.-H. Xiao, Chinese character recognition: History, status and prospects, Frontiers of Computer Science in China 1 (2) (2007) 126–136.

[2] H. Fujisawa, Forty years of research in character and document recognition—an industrial perspective, Pattern Recognition 41 (8) (2008) 2435–2446.

[3] Q.-F. Wang, F. Yin, C.-L. Liu, Handwritten Chinese text recognition by integrating multiple contexts, IEEE Trans. Pattern Analysis and Machine Intelligence 34 (8) (2012) 1469–1481.

[4] S. Katz, Estimation of probabilities from sparse data for the language model component of a speech recognizer, IEEE Trans. Acoustics, Speech and Signal Processing 35 (3) (1987) 400–401.

[5] S. F. Chen, J. Goodman, An empirical study of smoothing techniques for language modeling, in: Proc. 34th Annual Meeting on Association for Computational Linguistics, 1996, pp. 310–318.

[6] U.-V. Marti, H. Bunke, Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system, International Journal of Pattern Recognition and Artificial Intelligence 15 (1) (2001) 65–90.

[7] H. Bunke, S. Bengio, A. Vinciarelli, Offline recognition of unconstrained handwritten texts using HMMs and statistical language models, IEEE Trans. Pattern Analysis and Machine Intelligence 26 (6) (2004) 709–720.

[8] S. Espana-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya, F. Zamora-Martinez, Improving offline handwritten text recognition with hybrid HMM/ANN models, IEEE Trans. Pattern Analysis and Machine Intelligence 33 (4) (2011) 767–779.

[9] X.-D. Zhou, D.-H. Wang, F. Tian, C.-L. Liu, M. Nakagawa, Handwritten Chinese/Japanese text recognition using semi-Markov conditional random fields, IEEE Trans. Pattern Analysis and Machine Intelligence 35 (10) (2013) 2413–2426.

[10] A. Bissacco, M. Cummins, Y. Netzer, H. Neven, PhotoOCR: Reading text in uncontrolled conditions, in: Proc. ICCV, 2013, pp. 785–792.

[11] D.-H. Wang, C.-L. Liu, X.-D. Zhou, An approach for real-time recognition of online Chinese handwritten sentences, Pattern Recognition 45 (10) (2012) 3661–3675.

[12] Q.-F. Wang, F. Yin, C.-L. Liu, Unsupervised language model adaptation for handwritten Chinese text recognition, Pattern Recognition 47 (3) (2014) 1202–1216.

[13] R. Messina, J. Louradour, Segmentation-free handwritten Chinese text recognition with lstm-rnn, in: Proc. 13th Int. Conf. on Document Analysis and Recognition, 2015, pp. 171–175.

[14] B. Carpenter, Scaling high-order character language models to gigabytes, in: Proc. the Workshop on Software, 2005, pp. 86–99.

[15] Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin, A neural probabilistic language model, Journal of Machine Learning Research 3 (2) (2003) 1137–1155.

[16] H. Schwenk, Continuous space language models, Computer Speech & Language 21 (3) (2007) 492–518.

[17] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, S. Khudanpur, Recurrent neural network based language model., in: Proc. INTERSPEECH, 2010, pp. 1045–1048.

[18] H. Schwenk, Continuous space translation models for phrase-based statistical machine translation., in: Proc. COLING, 2012, pp. 1071–1080.

[19] H. Schwenk, A. Rousseau, M. Attik, Large, pruned or continuous space language models on a gpu for statistical machine translation, in: Proc. NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT, 2012, pp. 11–19.

[20] F. Zamora-Martínez, V. Frinken, S. España-Boquera, M. Castro-Bleda, A. Fischer, H. Bunke, Neural network language models for off-line handwriting recognition, Pattern Recognition 47 (4) (2014) 1642–1652.

[21] Y.-C. Wu, F. Yin, C.-L. Liu, Evaluation of neural network language models in handwritten Chinese text recognition, in: Proc. 13th Int. Conf. on Document Analysis and Recognition, 2015, pp. 166–170.

[22] A. Mnih, G. Hinton, Three new graphical models for statistical language modelling, in: Proc. 24th International Conference on Machine Learning, 2007, pp. 641–648.

[23] T. Morioka, T. Iwata, T. Hori, T. Kobayashi, Multiscale recurrent neural network based language model, in: Proc. INTERSPEECH, 2015, pp. 2366–2370.

[24] K. Irie, R. Schlüter, H. Ney, Bag-of-words input for long history representation in neural network-based language models for speech recognition, in: Proc. INTERSPEECH, 2015, pp. 2371–2375.

[25] A. Mnih, G. E. Hinton, A scalable hierarchical distributed language model, in: Proc. Advances in neural information processing systems, 2009, pp. 1081–1088.

[26] F. Morin, Y. Bengio, Hierarchical probabilistic neural network language model, in: Proc. AISTATS, Vol. 5, 2005, pp. 246–252.

[27] A. Mnih, Y. W. Teh, A fast and simple algorithm for training neural probabilistic language models, in: Proc. 29th International Conference on Machine Learning, 2012, pp. 1751–1758.

[28] T. Mikolov, S. Kombrink, L. Burget, J. H. Černockỳ, S. Khudanpur, Extensions of recurrent neural network language model, in: Proc. ICASSP, 2011, pp. 5528–5531.

[29] Y. Bengio, J.-S. Senecal, Adaptive importance sampling to accelerate training of a neural probabilistic language model, IEEE Trans. Neural Networks 19 (4) (2008) 713–722.

[30] T. Mikolov, A. Deoras, D. Povey, L. Burget, J. Černockỳ, Strategies for training large scale neural network language models, in: Proc. ASRU, 2011, pp. 196–201.

[31] S. Kombrink, T. Mikolov, M. Karafiát, L. Burget, Recurrent neural network based language modeling in meeting recognition., in: INTERSPEECH, 2011, pp. 2877–2880.

[32] S. Wang, L. Chen, L. Xu, W. Fan, J. Sun, S. Naoi, Deep knowledge training and heterogeneous CNN for handwritten Chinese text recognition, in: Proc. 15th ICFHR, 2016, pp. 84–89.

[33] C.-L. Liu, M. Koga, H. Fujisawa, Lexicon-driven segmentation and recognition of handwritten character strings for Japanese address reading, IEEE Trans. Pattern Analysis and Machine Intelligence 24 (11) (2002) 1425–1437.

[34] H. Lee, B. Verma, Binary segmentation algorithm for English cursive handwriting recognition, Pattern Recognition 45 (4) (2012) 1306–1317.

[35] X.-D. Zhou, J.-L. Yu, C.-L. Liu, T. Nagasaki, K. Marukawa, Online handwritten Japanese character string recognition incorporating geometric context, in: Proc. 9th Int. Conf. on Document Analysis and Recognition, 2007, pp. 48–52.

[36] F. Yin, Q.-F. Wang, X.-Y. Zhang, C.-L. Liu, ICDAR 2013 Chinese handwriting recognition competition, in: Proc. 12th Int. Conf. on Document Analysis and Recognition, 2013, pp. 1464–1470.

[37] D. Cireşan, U. Meier, Multi-column deep neural networks for offline handwritten Chinese character classification, in: Proc. IJCNN, 2015, pp. 1–6.

[38] C. Wu, W. Fan, Y. He, J. Sun, S. Naoi, Handwritten character recognition by alternately trained relaxation convolutional neural network, in: Proc. ICFHR, 2014, pp. 291–296.

[39] Z. Zhong, L. Jin, Z. Xie, High performance offline handwritten Chinese character recognition using GoogLeNet and directional feature maps, in: Proc. 13th Int. Conf. on Document Analysis and Recognition, 2015, pp. 846–850.

[40] X.-Y. Zhang, Y. Bengio, C.-L. Liu, Online and offline handwritten chinese character recognition: A comprehensive study and new benchmark, Pattern Recognition 61 (2017) 348–360.

[41] N. Li, X. Gao, L. Jin, Curved segmentation path generation for unconstrained handwritten Chinese text lines, in: Proc. APCCAS, 2008, pp. 501–505.

[42] L. Xu, F. Yin, Q.-F. Wang, C.-L. Liu, Touching character separation in Chinese handwriting using visibility-based foreground analysis, in: Proc. 11th Int. Conf. on Document Analysis and Recognition, 2011, pp. 859–863.

[43] J. H. Bae, K. C. Jung, J. W. Kim, H. J. Kim, Segmentation of touching characters using an MLP, Pattern Recognition Letters 19 (8) (1998) 701–709.

[44] L. Xu, F. Yin, Q.-F. Wang, C.-L. Liu, An over-segmentation method for single-touching chinese handwriting with learning-based filtering, International Journal on Document Analysis and Recognition 17 (1) (2014) 91–104.

[45] M. Nakagawa, Z. Bilan, M. Onuma, A model of on-line handwritten Japanese text recognition free from line direction and writing format constraints, IEICE transactions on information and systems 88 (8) (2005) 1815–1822.

[46] Y.-C. Wu, F. Yin, C.-L. Liu, Evaluation of geometric context models for handwritten numeral string recognition, in: Proc. ICFHR, 2014, pp. 193–198.

[47] T. Mikolov, A. Deoras, S. Kombrink, L. Burget, J. Cernockỳ, Empirical evaluation and combination of advanced language modeling techniques., in: INTERSPEECH, 2011, pp. 605–608.

[48] M. Sundermeyer, I. Oparin, J.-L. Gauvain, B. Freiberg, R. Schluter, H. Ney, Comparison of feedforward and recurrent neural network language models, in: Proc. ICASSP, 2013, pp. 8430–8434.

[49] M. Sundermeyer, H. Ney, R. Schluter, From feedforward to recurrent lstm neural networks for language modeling, IEEE/ACM Trans. Audio, Speech, and Language Processing 23 (3) (2015) 517–529.

[50] E. Arisoy, T. N. Sainath, B. Kingsbury, B. Ramabhadran, Deep neural network language models, in: Proc. NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT, Association for Computational Linguistics, 2012, pp. 20–28.

[51] L. Hai Son, A. Allauzen, F. Yvon, Measuring the influence of long range dependencies with neural network language models, in: Proc. NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT, Association for Computational Linguistics, 2012, pp. 1–10.

[52] N. Li, J. Chen, H. Cao, B. Zhang, P. Natarajan, Applications of recurrent neural network language model in offline handwriting recognition and word spotting, in: Proc. ICFHR, 2014, pp. 134–139.

[53] N. Li, J. Chen, H. Cao, B. Zhang, P. Natarajan, Applications of recurrent neural network language model in offline handwriting recognition and word spotting, in: Proc. ICFHR, 2014, pp. 134–139.

[54] C.-L. Liu, F. Yin, D.-H. Wang, Q.-F. Wang, CASIA online and offline Chinese handwriting Databases, in: Proc. 11th Int. Conf. on Document Analysis and Recognition, 2011, pp. 37–41.

[55] C.-L. Liu, Handwritten chinese character recognition: effects of shape normalization and feature extraction, in: Arabic and Chinese handwriting recognition, 2008, pp. 104–128.

[56] C. Liu, H. Fujisawa, Classification and learning in character recognition: advances and remaining problems, Machine Learning in Document Analysis and Recognition, S. Marinai and H. Fujisawa, eds 139–161.

[57] F. Yin, Q.-F. Wang, C.-L. Liu, Transcript mapping for handwritten Chinese documents by integrating character recognition model and geometric context, Pattern Recognition 46 (10) (2013) 2807–2818.

[58] C. M. Bishop, Pattern recognition and machine learning (2006) 225–284.

[59] J. Fürnkranz, A study using n-gram features for text categorization, Austrian Research Institute for Artifical Intelligence 3 (1998) 1–10.

[60] T. Joshua, J. Goodman, A bit of progress in language modeling extended version, Machine Learning and Applied Statistics Group Microsoft Research (2001) 1–72.

[61] J. Goodman, Classes for fast maximum entropy training, in: Proc. ICASSP, IEEE, 2001, pp. 561–564.

[62] G. Zweig, K. Makarychev, Speed regularization and optimality in word classing, in: Proc. ICASSP, 2013, pp. 8237–8241.

[63] P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, J. C. Lai, Class-based n-gram models of natural language, Computational linguistics 18 (4) (1992) 467–479.

[64] R. Kneser, H. Ney, Improved clustering techniques for class-based statistical language modelling., in: Proc. Eurospeech, 1993, pp. 973–76.

[65] T. Mikolov, S. Kombrink, A. Deoras, L. Burget, J. Cernocky, Rnnlm-recurrent neural network language modeling toolkit, in: Proc. ASRU, 2011, pp. 196–201.

[66] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.

[67] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[68] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.

[69] Z. Zhong, L. Jin, Z. Xie, High performance offline handwritten chinese character recognition using googlenet and directional feature maps, in: Proc. 13th Int. Conf. on Document Analysis and Recognition, 2015, pp. 846–850.

[70] C.-L. Liu, K. Marukawa, Pseudo two-dimensional shape normalization methods for handwritten chinese character recognition, Pattern Recognition 38 (12) (2005) 2242–2255.

[71] C.-L. Liu, H. Sako, H. Fujisawa, Effects of classifier structures and training regimes on integrated segmentation and recognition of handwritten numeral strings, IEEE Trans. Pattern Analysis and Machine Intelligence 26 (11) (2004) 1395–1407.

[72] X. He, Y.-C. Wu, K. Chen, F. Yin, C.-L. Liu, Neural network based over-segmentation for scene text recognition, in: Proc. ACPR, 2015, pp. 715–719.

[73] C.-L. Liu, Normalization-cooperated gradient feature extraction for handwritten character recognition, IEEE Trans. Pattern Analysis and Machine Intelligence 29 (8) (2007) 1465–1469.

[74] F. Kimura, K. Takashina, S. Tsuruoka, Y. Miyake, Modified quadratic discriminant functions and the application to Chinese character recognition, IEEE Trans. Pattern Analysis and Machine Intelligence 9 (1) (1987) 149–153.

[75] S. Yu, H. Duan, B. Swen, B.-B. Chang, Specification for corpus processing at Peking University: Word segmentation, pos tagging and phonetic notation., Journal of Chinese Language and Computing 13 (2).

[76] `http://www.bfsu-corpus.org/channels/corpus`.

[77] T.-H. Su, T.-W. Zhang, D.-J. Guan, H.-J. Huang, Off-line recognition of realistic Chinese handwriting using segmentation-free strategy, Pattern Recognition 42 (1) (2009) 167 – 182.

[78] A. Stolcke, Srilm - an extensible language modeling toolkit, in: Proc. INTERSPEECH, 2002, pp. 901–904.

[79] H. Schwenk, CSLM - A modular open-source continuous space language modeling toolkit., in: Proc. INTERSPEECH, 2013, pp. 1198–1202.

[80] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional architecture for fast feature embedding, arXiv preprint arXiv:1408.5093.

[81] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks., in: Proc. Aistats, Vol. 9, pp. 249–256.

[82] M.-K. Zhou, X.-Y. Zhang, F. Yin, C.-L. Liu, Discriminative quadratic feature learning for handwritten chinese character recognition, Pattern Recognition 49 (2016) 7–18.

[83] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, J. Schmidhuber, A novel connectionist system for unconstrained handwriting recognition, IEEE Trans. Pattern Analysis and Machine Intelligence 31 (5) (2009) 855–868.