Pattern Recognition 58 (2016) 190–203

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

DropSample: A new training method to enhance deep convolutional neural networks for large-scale unconstrained handwritten Chinese character recognition

Weixin Yang [a], Lianwen Jin [a,*], Dacheng Tao [b], Zecheng Xie [a], Ziyong Feng [a]

[a] *College of Electronic and Information Engineering, South China University of Technology, Guangzhou 510640, China*
[b] *Centre for Quantum Computation & Intelligent Systems, University of Technology, Sydney, Australia*

ARTICLE INFO

Article history:
Received 24 March 2015
Received in revised form
20 March 2016
Accepted 13 April 2016
Available online 23 April 2016

Keywords:
Convolutional neural network
Deep learning
Handwritten character recognition
Domain-specific knowledge

ABSTRACT

Inspired by the theory of Leitner's learning box from the field of psychology, we propose *DropSample*, a new method for training deep convolutional neural networks (DCNNs), and apply it to large-scale online handwritten Chinese character recognition (HCCR). According to the principle of *DropSample*, each training sample is associated with a quota function that is dynamically adjusted on the basis of the classification confidence given by the DCNN softmax output. After a learning iteration, samples with low confidence will have a higher frequency of being selected as training data; in contrast, well-trained and well-recognized samples with very high confidence will have a lower frequency of being involved in the ongoing training and can be gradually eliminated. As a result, the learning process becomes more efficient as it progresses. Furthermore, we investigate the use of domain-specific knowledge to enhance the performance of DCNN by adding a domain knowledge layer before the traditional CNN. By adopting *DropSample* together with different types of domain-specific knowledge, the accuracy of HCCR can be improved efficiently. Experiments on the CASIA-OLHDWB 1.0, CASIA-OLHWDB 1.1, and ICDAR 2013 online HCCR competition datasets yield outstanding recognition rates of 97.33%, 97.06%, and 97.51% respectively, all of which are significantly better than the previous best results reported in the literature.

1. Introduction

A traditional isolated online handwritten Chinese character recognition (HCCR) method typically employs the following framework: (1) pre-processing of an input handwritten character (e.g., linear or non-linear normalization [2] and addition of imaginary strokes [3–7]), (2) feature extraction (e.g., 8-directional feature extraction [6] and discriminative directional feature extraction [8]), and (3) classification via machine learning methods (e.g., modified quadratic discriminant function (MQDF) [9,10], support vector machine [11], and hidden Markov model (HMM) [12]). In contrast, deep learning methods [13–17], which have attracted a considerable amount of research and industry attention in recent years, deviate from the above-mentioned framework by providing an alternative end-to-end solution to HCCR without any dedicated feature extraction or pre-processing technique, enabling potentially high performance. Owing to the availability of large-scale training data, new training technologies (e.g., Dropout [18], DropConnect [19], layer-wise pre-training [20]), and advanced computing hardware platforms (e.g., GPU [21]), the convolutional neural networks (CNNs) originally proposed by LeCun in the 1990s [22,23] have been extensively investigated in recent years. The traditional CNN has been extended with deeper architectures (e.g., [24,25]; we refer to this variant of CNN as Deep CNN (DCNN) in this paper), advanced training technologies, and effective learning algorithms (e.g., [18,40]) to address the various challenges posed by computer vision and pattern recognition problems. Consequently, significant breakthroughs have been achieved, such as image recognition [24–27], facial recognition [28,29], handwriting recognition [13,15,18,30], pose recognition [31], text detection, and natural scene image recognition [32–36]. Furthermore, DCNN with different structures has been successfully applied to the field of HCCR field [13–17], which poses a major challenge because it involves a large vocabulary (e.g., as many as 3755 classes for the GB2312-80 level-1 standard), many similar and confusable characters, and different writing styles with unconstrained cursive techniques [37].

The performance of current DCNNs is highly dependent on the greedy learning of model parameters via many iterations on the basis of a properly designed network architecture with abundant labeled training data. Most DCNN models treat all the training

* Corresponding author. Tel.: +86 13802994591.
E-mail address: lianwen.jin@gmail.com (L. Jin).

http://dx.doi.org/10.1016/j.patcog.2016.04.007
0031-3203/© 2016 Elsevier Ltd. All rights reserved.

Table 1
Explanation of several psychological terminologies mentioned in this paper.

Terminology	Interpretation
Forgetting curve	It describes the exponential loss of information that one has learned [64]
Recency effect	It describes the increased recall of the most recent information because it is still in the short-term memory [64]
Spacing effect	It is the phenomenon whereby animals more easily remember of learned items when they are studied a few times spaced over a long time span rather than repeatedly studied in a short span of time [64]
Spacing repetition	It is a learning technique that incorporates increasing intervals of time between subsequent reviews of previously learned material in order to exploit the psychological spacing effect [39]

samples uniformly during the entire learning process. However, we have found that the error reduction rate during the learning process is initially high but decreases after a certain number of training iterations. This may be attributed to most of the samples being well recognized after a certain number of training iterations; thus, the error propagation for adapting the network parameters is low, while confusable samples, which are difficult to learn and account for a relatively low ratio of the training dataset, do not have a high likelihood of contributing to the learning process. Some previous DCNNs consider this phenomenon as a signal to manually reduce the learning rate, or as a criterion for early stopping [38], thereby neglecting the potential of the confusing samples that have thus far been insufficiently well-learnt.

Inspired by the theory of Leitner's learning box from the field of psychology [1], we propose a new training method, namely *DropSample*, to enhance the efficiency of the learning process of DCNN, and we employ it to solve the challenge of cursive online HCCR. The explanation of several relevant psychological terminologies are presented in Table 1. The principle of spacing repetition is useful in many contexts, but it requires a learner to acquire a large number of items and retain them in memory indefinitely. Leitner's learning box is designed as a simple implementation of the principle of spacing repetition for learning [39]. A direct application of Leitner's learning box theory is that material that is difficult to learn will appear more frequently and material that is easy to learn will appear less frequently, with difficulty defined according to the ease with which the user is able to produce a correct learning response. The *DropSample* training method proposed in this paper adopts a similar concept to design a learning algorithm for DCNN. To this end, each training sample is assigned to a box with a quota function that is dynamically adjusted according to the classification confidence given by the DCNN softmax output. After a learning mini-batch iteration, samples with high confidence in this mini-batch will be placed in a box with low appearance frequency, whereas those with low confidence will be placed in another box with high appearance frequency, thus they are more likely to appear for selection as the training data after a short learning interval. A certain amount of noisy data (e.g., mislabeled samples and outliers) always exists in the training dataset, and such data may be useful in the initial stages of training to avoid overfitting; however, they gradually prevent the network from achieving high prediction accuracy. They should therefore be placed in a box with low appearance frequency, and thus can gradually be eliminated. A schematic of *DropSample* is shown in Fig. 1.

To address the specific challenge of online HCCR, we propose the incorporation of domain-specific technologies in DCNN models to account for domain-specific information that may be useful but cannot be learnt by the DCNN. By employing the new training technology of *DropSample* together with various types of domain-specific knowledge, we find that recognition accuracy can be improved significantly. In this way, we obtain a single network that achieves test error rates of 2.99% and 3.43% on CASIA-OLHWDB 1.0 and CASIA-OLHWDB 1.1 [41], respectively, both of which are lower than the state-of-the-art results reported in previous studies [13,15,37]. Furthermore, the different types of domain-specific knowledge contribute to corresponding DCNN classifiers, which may be complementary and can therefore be integrated to achieve better accuracy. The ensemble ultimately reduces the test error rates to 2.67% on CASIA-OLHWDB 1.0, 2.94% on CASIA-OLHWDB 1.1, and 2.49% in the case of the ICDAR 2013 online HCCR competition dataset [16], which are significantly better than the best results reported in previous studies [13,15,16,37]; this confirms the effectiveness of the proposed *DropSample* training method.

The remainder of this paper is organized as follows. Section 2 introduces related studies reported in the literature. Section 3 presents the proposed DCNN architecture and the configurations employed. Section 4 provides a detailed description of the *DropSample* training method. Section 5 describes the domain-specific knowledge. Section 6 presents the experimental results and analysis. Lastly, Section 7 summarizes our findings and concludes the paper.

2. Related work

Over the past four decades, numerous studies have investigated HCCR [42], resulting in the development of many techniques such as non-linear normalization, data augmentation, directional feature extraction, and quadratic classifier modification. Non-linear normalization methods such as modified centroid-boundary alignment (MCBA) [43], line density projection interpolation (LDPI) [2], and line density projection fitting (LDPF) [43], as well as their pseudo-two-dimensional extensions [2,44], are based on line density equalization that can reduce within-class variation of character shape. Data augmentation techniques enhance insufficient training data by generating various handwriting styles via affine transformation [45], cosine functions [46], distorted generation [47], deformation transformation [48], and style consistent perturbation [57]. Two of the most popular feature extraction methods for representation are 8-directional feature extraction [6,10] and discriminative directional feature extraction [8,49]. By employing classifiers that use the modified quadratic discriminant function (MQDF) [9,10], or its variant such as discriminative learning quadratic discriminant function (DLQDF) [37], a traditional HCCR system can yield fairly good recognition performance.

In recent years, CNNs have produced outstanding results in the fields of machine learning and pattern recognition. The idea of CNN was first proposed by Fukushima [50] in 1980. It was formally developed by LeCun et al. [22,23] and improved by Simard et al. [51], Cireşan et al. [13,14], and others. GPU acceleration hardware [21] has facilitated the development of deep CNN (DCNN), which includes a deeper architecture with additional convolutional layers. DCNN offers several advantages. For example, it enables integrated training of feature extractors and classifiers to provide systematic optimization. In addition, it does not require pre-training and provides useful properties such as availability of raw data, effective feature extraction, and excellent generalization capability [52].

CNN has acquired a reputation for solving many computer vision problems in recent years, and its application to the field of HCCR has been shown to provide significantly better results than traditional methods [13–17]. The multi-column deep neural network (MCDNN) method proposed by Cireşan et al. [13] shows remarkable ability in many applications and attains near-human performance on handwritten datasets such as MNIST. In addition, it has provided promising results for HCCR. Graham proposed a variation of CNN called DeepCNet [15], which won first place at the ICDAR 2013 online HCCR competition [16]. By combining the path signature feature and employing a spatially sparse architecture, DeepCNet produced a best test error rate of 3.58% [15] on CASIA-OLHWDB 1.1, which is lower than that achieved by MCDNN (5.61%) [13] and DLQDF (5.15%) [37].

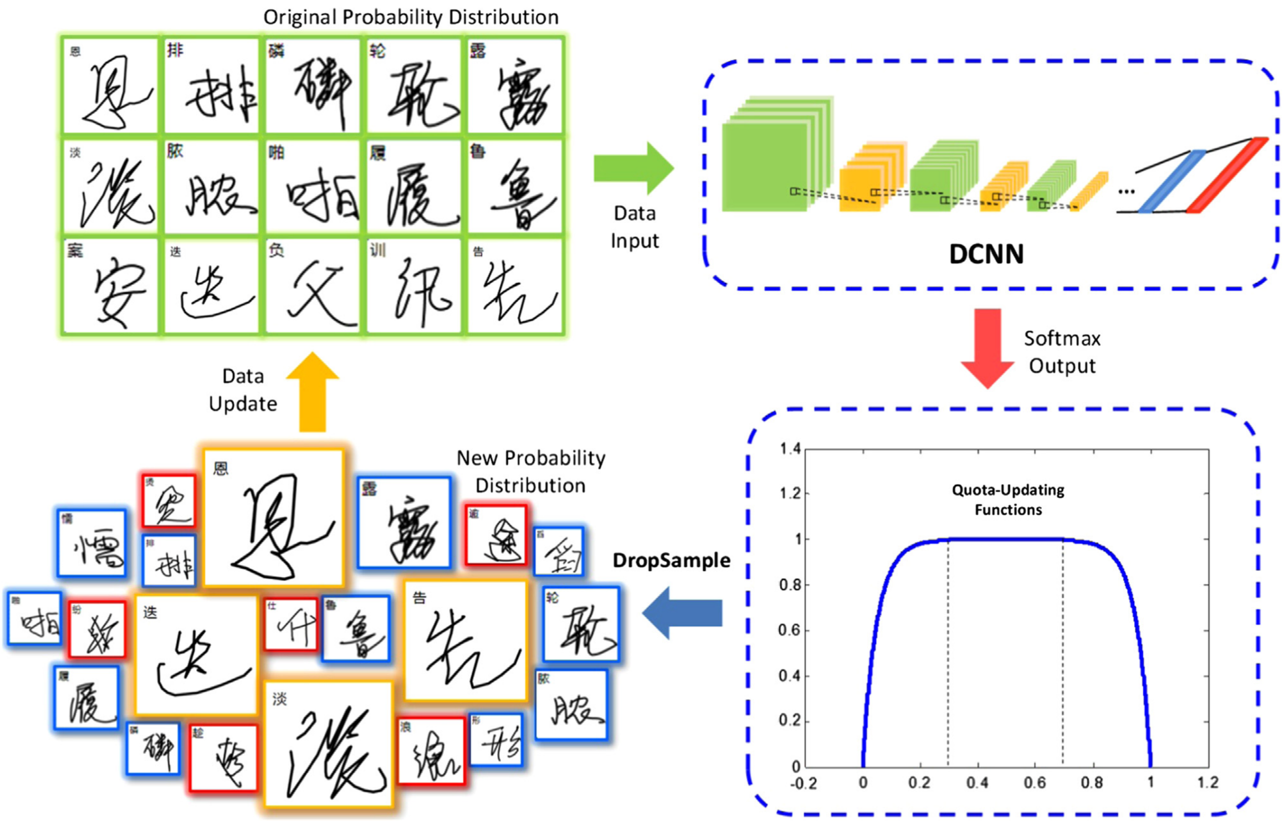

Fig. 1. Illustration of *DropSample*. Initially, an uniform probability (also called quota) is selected for each training sample (upper left). The DCNN softmax outputs are used to separate the training set into three groups (blue for well-recognized samples, yellow for confusing samples, and red for heavily noisy or mislabeled samples). The selected probability distributions are then updated by the quota-updating functions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Furthermore, the use of the deep model for the recognition of digits [22] and various texts of different languages, such as English [53], Arabic [54], Tibetan [55], Hangul [52], and Devanagari [56], has also attracted considerable attention.

3. Architecture of sparse DCNN

DeepCNet [15], the variation of CNN proposed by Graham as mentioned above, exploits the spatial sparsity of the input layer to reduce computational complexity. The spatial sparsity means that an input online isolated character drawn as an $N \times N$ binary image shows sparse pen-color pixels compared to its background pixels. The computational complexity of the image and trajectory is $O(N^2)$ and $O(N)$, respectively; therefore, it is reasonable to set rules to reduce the calculation of the background pixels. On the other hand, because convolutional networks often encounter the issue of spatial padding, DeepCNet provides a solution that adds a substantial amount of padding to the input image at no additional calculation cost for the sparsity. Moreover, DeepCNet employs slow convolutional and max-pooling layers instead of fast layers in its architecture. Smaller filters and deeper layers are used to gradually reduce the size of the feature maps so that more spatial information can be retained to improve the generalization.

The DCNN employed in this study offers the advantages described above, and its structure is similar to that of DeepCNet employed in [15]. However, our DCNN model differs from DeepCNet [15] in the three aspects: (1) We apply a domain-specific knowledge layer immediately after the input layer to make use of the domain knowledge that is useful for HCCR but cannot be learnt by the CNN. Such domain knowledge includes 8-directional feature maps [6], character maps with both real and imaginary strokes [7], the deformation transformation of handwritten Chinese characters [48], and path signature feature maps [15,58]. (2) We add an extra fully connected (FC) layer before the final softmax layer of CNN. (3) Our model is much slimmer than the DeepCNet model used for HCCR in [15]; in other words, we use a smaller number of convolutional kernels in each layer, and thus, fewer parameters have to be learnt and stored.

As shown in Fig. 2, the basic structure of our DCNN consists of five convolutional layers and two fully connected layers, while each of the convolutional layers is followed by a max-pooling layer. The size of the convolutional filter is 3×3 for the first layer and 2×2 for subsequent layers, with a stride of 1 pixel. Max-pooling is carried out over a 2×2 pixel window, with a stride of 2 pixels. Lastly, a stack of convolutional layers is followed by two FC layers containing 480 and 512 neurons, respectively. The number of convolutional filter kernels used is much smaller than is used in [15]; it is set as 80 for the first layer and increased in steps of 80 after each max-pooling. Consequently, the total number of parameters of our model is only 3.8 million, which is much smaller than the 5.9 million used by DeepCNet [15].

Rectified linear units (ReLUs) [59] are used as activation functions for neurons in all convolutional layers and FC layers. Compared to some saturating nonlinearities such as hyperbolic tangent [60] and sigmoid functions, non-saturating ReLUs can overcome the gradient diffusion problem and show better fitting ability when large-scale datasets such as handwritten Chinese characters are used for training [61]. In the output layer, the classification results are given by a k-way softmax layer, which outputs a probability distribution over the k classes. Note that the softmax output can be treated as both an indicator of classification results and a confidence measurement of a given input sample [40,71]. The confidence measurement is discussed in detail in Section 4.

First, we render the input handwritten Chinese character image as a 48×48 bitmap image, then we embed the image in a 96×96

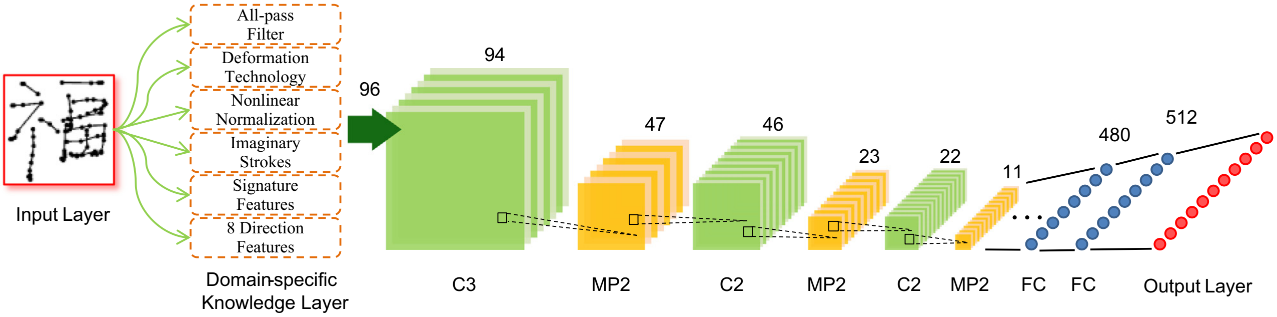

Fig. 2. Illustration of the basic DCNN for HCCR.

grid that is initialized to zero for spatial padding. The baseline architecture of our DCNN can be represented as

96×96Input-$M\times96\times96$-80C3-MP2-160C2-MP2-240C2-MP2-320C2-MP2-400C2-MP2-480FC-512FC-Output,

where M denotes the number of input channels, which varies from 1 to 30 depending on the different types of domain knowledge used.

4. *DropSample*: a new training method for DCNN

4.1. Motivation

During the training process of DCNN for large-scale pattern recognition tasks, we usually encounter three problems.

First, although the problem of heavy computation in DCNN training can be alleviated using GPU-based massive parallel processing [51], DCNN training with an extremely large amount of training data for large-scale classification remains a time-consuming process. This is because the DCNN has to learn millions of parameters, and the convolution produces a large number of additional computations compared with traditional fully-connected shallow networks [22].

Second, the training error reduction during the training process is initially high but decreases after a certain number of training epochs (e.g., after 5–10 training epochs). This may be attributed to most of the samples being well recognized after a certain number of training epochs; thus, the error propagation for adapting the network parameters is low, while confusable samples, which are difficult to learn and account for a relatively low ratio of the training set, do not have a high likelihood of being selected for the training process. Some DCNNs treat this phenomenon as a signal for early stopping [38], thereby neglecting the potential of the confusable samples that have not thus far been sufficiently well-learnt.

Third, despite extensive efforts being devoted to the dataset collection and cleaning, many databases include a significant number of mislabeled samples and heavily noisy samples. Fig. 3 shows some examples of heavily noisy samples or mislabeled samples from the CASIA-OLHWB dataset [41]. Although the mislabeled or noisy samples constitute a minority of the overall dataset, they should not be tolerated. A primary reason is that after many training epochs, the error reduction rate gradually decreases and starts to oscillate. Thus, mislabeled or heavily noisy samples might be harmful because a strong error feedback produced by these samples will back-propagate from the output layer to previous layers and interfere with the entire network.

To overcome the first two problems, LeCun et al. [62] proposed a viewpoint that networks learn the fastest from the most unexpected sample which offers a relatively large error from output. This method is enlightening but it takes an unavoidable and huge time consumption for detection of the samples with the largest error at each iteration, especially dealing with large-scale dataset. In addition, the noisy samples will gradually dominate the learning process and prevent the network from reaching local minimum. Yuan et al. [63] developed a simple error samples reinforcement learning (ESRL) algorithm, which aims to make several interruptions during the training process and reconstruct the training set by randomly abandoning well-recognized samples and selecting several complex samples for data augmentation. Although this algorithm seems useful, it requires manual interruption, and the improvement is limited. In addition, the reconstruction of the sample distribution is delayed, and the abandoned well-recognized samples are no longer recalled for the training process. Moreover, the interference due to heavily noisy samples might be magnified during data augmentation.

The method we will present in this paper is inspired from the field of psychology. Hermann Ebbinghaus, a German psychologist, discovered the forgetting curve, which describes the exponential loss of information that one has learned [64]. Inspired by his findings that the curve for meaningful material shows a slower decline than the curve for “nonsense” material, Sebastian Leitner proposed the Leitner system [1], which came to be widely used in learning. In this system, a set of flashcards is sorted into groups according to how well the learner knows each flashcard. If the learner succeeds in recalling the solution written on a flashcard, the card is sent to the next group; otherwise, it is sent back to the very first group. For each successive group, the frequency with which the learner is required to revisit the cards decreases [1]. Inspired by this concept, we thereby propose a new DCNN training method, called *DropSample*, to address the three problems mentioned above. A basic assumption of *DropSample* is that deep neural networks can have the “forgetting” mechanism somehow like human. We carried out an experimental test (reported in Section 6) to support this idea and found that for the classification task, the forgetting curve of a deep neural network is similar to that of human brain. Thus, it is reasonable to design the *DropSample* algorithm to mimic human learning process for efficient DCNN training. *DropSample* sorts the samples in each mini-batch into groups and uses an updating function to dynamically and automatically update the selected quotas of samples in different groups. The concept of quota-updating is somehow similar to that of the weight updating of samples in Adaboost [72], as both of them iteratively exploit the feedback of classifier to automatically improve the utility of data. In *DropSample*, the feedback which derives from the output layer with k-way softmax as the activation function, is a probability distribution over the n classes. According to a previous study, the softmax output of CNN can be regarded as a confidence measurement of the CNN classification output [40,71]. In the case of training using *DropSample*, samples with high confidence are placed in a box with low appearance probability and those with very low confidence are discarded so that the remaining samples can be frequently reviewed in the network.

4.2. Analysis

Suppose that we have m training samples (x_i, y_i). The input vectors are $(n+1)$-dimensional. Batch gradient descent is used to train a

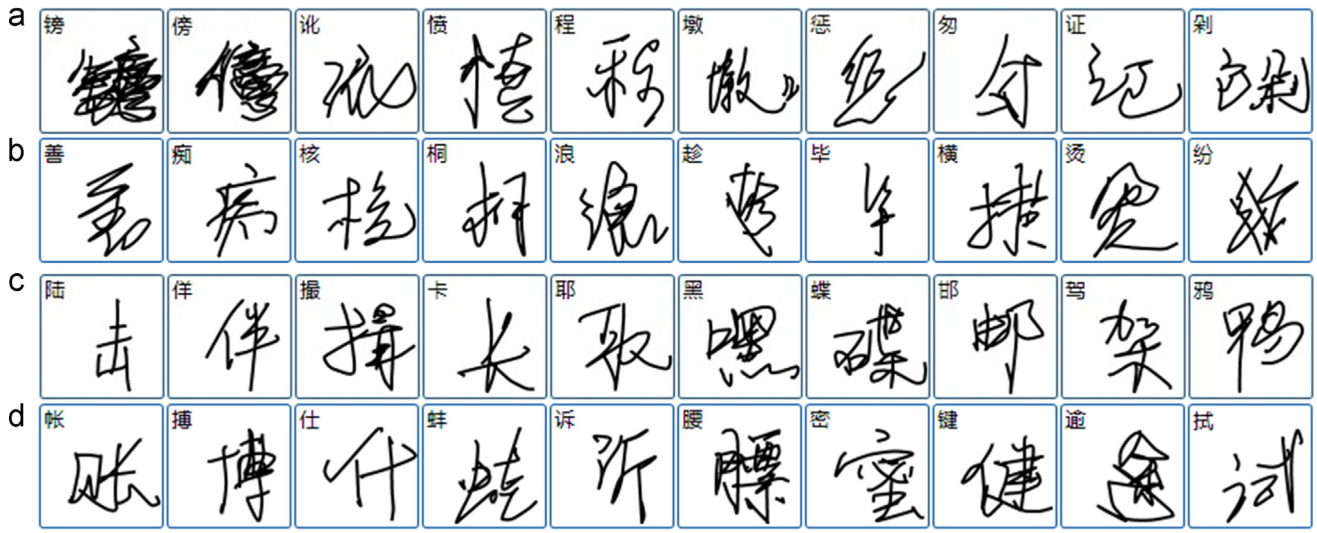

Fig. 3. Examples of heavily noisy samples, mislabeled samples, and outliers from the CASIA-OLHWDB dataset. The given label of each example is shown in the upper left corner. (a) and (b) comprise heavily noisy samples or outliers, whereas (c) and (d) comprise mislabeled samples or outliers. Examples of (a) and (c) are taken from CASIA-OLHWDB 1.0, whereas (b) and (d) are from CASIA-OLHWDB 1.1.

DCNN with k-way softmax in the output layer. The hypotheses of softmax can be written as

$$h_{\theta_j}(x_i) = \frac{\exp(\theta_j^T x_i)}{\sum_{l=1}^{k}\exp(\theta_l^T x_i)} \quad (j = 1, 2, \ldots, k), \tag{1}$$

where θ_j denotes the weights and bias corresponding to the jth output. The loss function $J(\theta)$ with a regularization term can be written in the form of cross entropy as

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\{y_i = j\}\log\frac{\exp(\theta_j^T x_i)}{\sum_{l=1}^{k}\exp(\theta_l^T x_i)}\right] + \frac{\lambda}{2}\sum_{i=1}^{k}\sum_{j=0}^{n}\theta_{ij}^2. \tag{2}$$

Its partial derivative (known as the “error term”) with respect to θ_j is given by

$$\nabla_{\theta_j} J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[x_i(1\{y_i = j\} - p(j|x_i;\theta))\right] + \lambda\theta_j, \tag{3}$$

where p is the probability distribution of the softmax output, $1\{\cdot\}$ is the indicator function, and λ is a penalty factor. Thus, one gradient descent iteration updates the parameters as

$$\theta_j := \theta_j - \alpha\nabla_{\theta_j} J(\theta) \quad (j = 1, 2, \ldots, k), \tag{4}$$

where α is the learning rate of the parameters.

According to (3) and (4), the error term can be split into three groups based on the softmax output. Let p_i be the probability of the labeled class of the i_{th} sample. Let T_1 and T_2 be the thresholds for p_i that roughly separate m training samples into three groups, M_1, M_2, and M_3, corresponding to the well-recognized group $(T_2 < p_i \le 1)$, confusing group $(T_1 \le p_i \le T_2)$, and noisy group $(0 \le p_i < T_1)$, respectively. Thus, (3) can be rewritten as

$$\begin{aligned}\nabla_{\theta_j} J(\theta) &= E_1 + E_2 + E_3 + \lambda\theta_j \\ &= -\frac{1}{m}\Bigg\{\sum_{i_1 \in M_1}\left[x_{i_1}(1\{y_{i_1} = j\} - p(j|x_{i_1};\theta))\right] \\ &\quad + \sum_{i_2 \in M_2}\left[x_{i_2}(1\{y_{i_2} = j\} - p(j|x_{i_2};\theta))\right] \\ &\quad + \sum_{i_3 \in M_3}\left[x_{i_3}(1\{y_{i_3} = j\} - p(j|x_{i_3};\theta))\right]\Bigg\} + \lambda\theta_j,\end{aligned} \tag{5}$$

where E_1, E_2, and E_3 represent the sub-error terms corresponding to the three above-mentioned sample groups, respectively, which are analyzed as follows.

(a) Well-recognized group M_1: Given that the values of the probability distribution p of a well-recognized sample are similar to those of the indicator function, the parameter θ_j obtains a small update from the sub-error term E_1 according to (4) and (5). Well-recognized samples are less useful for improving the network because they produce very low error feedback; therefore, it is reasonable to reduce their opportunities for selection as training data in the next training iteration.

(b) Confusing group M_2: In contrast, a confusing sample has a relatively decentralized softmax probability distribution. Neither the probability value of the labeled class nor the probability values of similar classes can be ignored, because both have an obvious effect on E_2. Moreover, in the feature space, confusing samples often appear near the decision boundary and exert an influence on boundary optimization. Therefore, the adaptive quotas of the confusing samples, which reflect their opportunities for selection as training data, should be increased to achieve fast, reinforced learning. However, learning too fast may bring extra problems. Frequently revisiting the confusing samples leads to increasing sensibility of noises in this group. That's why we concern the noisy group followed.

(c) Noisy group M_3: It is sometimes useful to acquire a large amount of feedback back-propagated from the sub-error terms [62]. However, an exception arises in the case of noisy samples, which often produce a larger sub-error term E_3 than E_1 or E_2. Three kinds of noisy samples are considered in this paper: outliers, mislabeled samples and confusing noisy samples. Outliers and mislabeled samples can be gradually excluded from the training set depending on the outputted confidant scores, but the confusing noisy samples, which belong to the confusing group during early training epochs, are difficult even for human to absolutely distinguish. Empirically, we use the difference (denoted as δ) of softmax output probabilities between the predicted class and the labeled class to discriminate the noisy samples from others in the confusing group, expressing as

$$\delta_i = \hat{p}_i - p_i, \quad (0 < \delta_i \le 1), \tag{6}$$

where $\hat{p}_i = \underset{j}{\operatorname{argmax}}\, p(j|x_i;\theta)$is the output probability of the predicted class, $p_i = p(y_i|x_i;\theta)$ is the output probability of the labeled class and i is the index of sample. Notice that the δ in (6) is positive, so we only consider the noisy samples located at the wrong sides of the current decision boundaries. Samples with smaller δ, which are closer to the boundaries, belong to the confusing group M_2, while those with larger δ can be more confidently regarded as the noises among the confusing samples. In the training process, all kinds of noisy samples would better be excluded from the training set, but not initially, as they are regarded as valuable noise for enhancing the regularization of the DCNN at the beginning of training [40]. It is also difficult for the neural network to identify noisy samples during early training epochs.

4.3. Implementation of DropSample

Each sample in the training dataset is equally allocated an initial quota fixed to 1, and an updating function is then defined to change this quota according to the softmax output of network. The softmax output here is considered as the performance evaluation of a certain sample [40], while the quota represents the frequency evaluation of it being selected into a training mini-batch. A difference between them is that the softmax output is instantaneously given by the current network, while the quota should be gradually adjusted on the basis of not only the current performance but also the previous quota, for example, a well-recognized sample has a low quota (i.e., less frequency) resulting from excellent performance every time it is selected. Thus, we develop a quota-updating strategy as

$$q_i^t = q_i^{t-1} f(p_i^t), \tag{7}$$

where $f(\cdot)$ denotes the quota-updating function, p_i^t is the softmax probability of the labeled class given by the ith sample at the tth updating mini-batch, and q_i^t is the adaptive quota of the ith sample at the tth updating mini-batch. At each mini-batch, the update only happens to the quotas of this mini-batch rather than all the quotas in the batch (i.e., the whole training set). The reason are two folds. First, mini-batch updating is usually much faster than batch updating, because during each iteration, the former only has a small amount of quotas updated, while the latter takes a huge time which is proportional to the size of the training set. Second, mini-batch updating provides a gentle adjustment of learning mechanism for the network to revisit samples and forget the well-performed one gradually, while batch updating is much more aggressive that it may ignore well-performed samples before they were sufficiently trained and therefore is more sensitive to noisy data.

In practical application, the function $f(\cdot)$ can be manually designed according to certain requirements or tasks. Here, we present an exponential function and a multi-level hierarchical function. The exponential piecewise function $f_1(\cdot)$ is given by

$$f_1(p_i^t) = \begin{cases} 1 - \exp(-\alpha p_i^t) & 0 \le p_i^t < T_1 \\ 1 - \exp(-\beta(1 - 2T_1 - \delta_i^t)) & T_1 \le p_i^t \le T_2 and \Delta \le \delta_i^t \le 1 - 2T_1 \\ 1 - \exp(-\gamma(1 - p_i^t)) & T_2 < p_i^t \le 1 \\ 1/q_i^{t-1} & otherwise \end{cases}, \tag{8}$$

where α, β and γ are the slope factors. A higher value of these slope factors indicates a steeper shape of the function. The parameter Δ is a threshold of δ; the δ larger than Δ is regarded as noisy samples as described in Section 4.2.

The multi-level hierarchical piecewise function $f_2(\cdot)$ is designed as

$$f_2(p_i^t) = \begin{cases} a_{1h} & \{L_{1h} \le p_i^t \le U_{1h} \mid 0 \le L_{1h} < U_{1h} < T_1, h = 1, 2, \ldots l_1\} \\ a_{2h} & \{T_1 \le p_i^t \le T_2 and L_{2h} \le \delta_i^t \le U_{2h} \mid \Delta \le L_{2h} < U_{2h} \le 1 - 2T_1, h = 1, 2, \ldots l_2\} \\ a_{3h} & \{L_{3h} \le p_i^t \le U_{3h} \mid T_2 < L_{3h} < U_{3h} \le 1, h = 1, 2, \ldots l_3\} \\ 1/q_i^{t-1} & otherwise \end{cases}, \tag{9}$$

where L_h and U_h denote the lower and upper boundary, respectively, of the hth level of the hierarchical function. The parameters a_{1h}, a_{2h}, and a_{3h} are the updating factors of the hth level. In both (8) and (9), the first two expressions apply to noisy group (M_3), the third expression performs on well-recognized group (M_1), and the last one relates to confusing group (M_2). Because M_2 is the group of confusing samples, we fix the corresponding expressions to $1/q_i^{t-1}$ to maintain the value of 1 for the quotas.

In our implementation, the parameters for $f_1(\cdot)$ in (8) are determined empirically through experiments on our training data. The details are presented as follows: α=400, β=10, γ=600, T_1=1/k, T_2=0.99 and Δ=0.05. The parameters for $f_2(\cdot)$ in (9) are set as follows: l_1=l_2=l_3=3, a_{11}=a_{21}=a_{31}=0.9, a_{12}=a_{22}=a_{32}=0.5, and a_{13}=a_{23}=a_{33}=0.3. The boundaries are L_{11}=0, U_{11}=L_{12}=1/4k, U_{12}=L_{13}=1/2k, U_{13}=T_1=1/k; L_{21}=Δ=0.20, U_{21}=L_{22}=0.40, U_{22}=L_{23}=0.60, U_{23}=1-2T_1; T_2=L_{31}=0.99, U_{31}=L_{32}=0.999, U_{32}=L_{33}=0.9999, and U_{33}=1.

The threshold T_1, which defines the boundary of outliers and mislabeled samples, is set to 1/k because this value is equivalent to the random guess probability of k classes. We consider a sample to be heavily noisy if the probability of its labeled class is lower than this threshold after a certain number of mini-batch training iterations (300,000 in our implementation, to ensure a fair judgment). Because we cannot strictly distinguish heavily noisy samples, the random guess probability ensures that samples below this threshold are neither absolutely correct labeled samples nor characters similar to the true labeled sample. The threshold T_2, which defines the boundary of well-recognized samples, should be set to a high value close to 1.

Algorithm 1 *DropSample* training algorithm

```
Input: training set $X = \{(x_i, y_i)\}, i = 1, \ldots, m$of $k$ classes.
Output: network parameters $\theta$.
Initialization: iteration $t \leftarrow 0$; learning rate$\alpha(t)$; quota parameters$q_i^0 \leftarrow 1, \forall i$;
quota-updating function$f_1$; $\alpha = 400$, $\beta = 10$, $\gamma = 600$, $T_1 = 1/k$, $T_2 = 0.99$, $\Delta = 0.20$.
while not converge do
    $P^t = (q_1^t/Z^t, \ldots, q_m^t/Z^t)^T$   where$Z^t = \sum_{i=1}^m q_i^t$
    $t \leftarrow t+1$ sample a mini-batch from $X$ based on$P^t$
    Forward propagation: get the softmax output of the labeled class$p_i^t$
    Back propagation: calculate error term $\nabla_\theta J(\theta) = E_1 + E_2 + E_3 + \lambda\theta$
    Update network parameters $\theta = \theta - \alpha(t)\nabla_\theta J(\theta)$
    Calculate quota-updating function$f_1(p_i^t)$
      if $0 \le p_i^t < T_1$,$f_1(p_i^t) = 1 - \exp(-\alpha p_i^t)$
      else if $T_1 \le p_i^t \le T_2$and$\Delta \le \delta_i^t \le 1 - 2T_1$ , $f_1(p_i^t) = 1 - \exp(-\beta(1 - 2T_1 - \delta_i^t))$
      else if $T_2 < p_i^t \le 1$,$f_1(p_i^t) = 1 - \exp(-\gamma(1 - p_i^t))$
      else $f_1(p_i^t) = 1/q_i^{t-1}$
    Update quota parameters $q_i^t \leftarrow q_i^{t-1} f_1(p_i^t)$
end while
```

Finally, the proposed *DropSample* training algorithm is summarized in Algorithm 1. It is worth to note that in line 4 and line 7 of the algorithm, the quota-updating depends on the network's output before back propagation. The main reason is as follows. During a mini-batch training, the samples just trained might be shown temporarily but misleadingly excellent accuracy by the network after back propagation according to the recency theory in psychology [39,64]. Another reason is that extra time of additional test is need to obtain the latest testing error after back propagation. As the quotas formed within a period of time are somewhat robust to the current performance, it is time-saving to use the error before back propagation rather than after it.

5. Domain-specific knowledge layer of DCNN

Our DCNN architecture consists of a domain-specific knowledge layer that is used to process and embed useful domain knowledge, as shown in Fig. 2. Previous studies have shown that the incorporation of domain-specific knowledge, such as path signature feature maps [15], is very useful for achieving highly successful online HCCR. This inspired us to adopt various types of domain knowledge in our DCNN models in this study, including deformation transformation [47,48], non-linear normalization [2], imaginary stroke maps [3], 8-directional feature maps [9], and path signature feature maps [58].

5.1. Deformation transformation

Deformation transformation (DT) for DCNN is used to provide shape variation and generate a substantial amount of online training data. It is very helpful for enhancing the performance of DCNN because a distribution that is considerably more representative can be roughly simulated by elaborately designing the transformation and applying it to the training samples in order to enhance the generalization capability of the network. In DeepCNet [15], the training set is extended by affine transformation, including global stretching, scaling, rotation, and translation, but only stroke jiggling is used for local deformation. To enrich the data with local diversity, two deformation methods are considered in this paper. The first is the distorted sample generation method [47], which can handle both shearing and local resizing. The second is deformation transformation [48], which provides adjustable parameters for shrinking or stretching parts of the character both locally and globally to generate rich handwriting styles.

5.2. Non-linear normalization

The non-linear normalization (NLN) method is based on line density equalization and allows shape correction, which has been shown to be a crucial pre-processing technique for traditional HCCR methods. In our implementation, we use an NLN method called line density projection interpolation (LDPI), which has been reported to show good performance in a previous study [2]. However, given that the max-pooling in DCNN can absorb positional shifts, the NLN step might be redundant or even result in information loss [52]. Moreover, the NLN step for data pre-processing may cause loss of diversity and reduce the generalization capability of the network. Therefore, our DCNN incorporates the NLN step before deformation transformation in the domain knowledge layer in order to retain the diversity of the training samples.

5.3. Imaginary strokes technique

Imaginary strokes (IS) [3] are those pen-moving trajectories in pen-up states that are not recorded in the original character sample, and experiments have shown that they facilitate effective HCCR [3–7]. An imaginary strokes map is defined as a character map that consists of all the straight lines from the end point of each pen-down stroke to the start point of the next pen-down stroke [6]. It is extremely difficult for a DCNN to learn such virtual information only from a 2D character map. Moreover, a common drawback of adding such virtual information to the prototype is that similarities may be introduced between readily distinguishable characters. Thus, a trade-off between imaginary strokes and original strokes is recommended, in which case the DCNN performs well. Therefore, our DCNN incorporates both real stroke maps and imaginary stroke maps in the domain knowledge layer. This approach is found to be useful even though the number of feature maps in the domain knowledge layer is doubled.

5.4. Path signature feature maps

Path signatures (Sign.), in the form of iterated integrals, was developed in recent years by Lyons as a fundamental role in the rough theory [58] and it can be used to solve any linear differential equation and uniquely express a finite path. The path signature feature was first applied to the recognition of online handwritten characters by Graham [15]. In general, the first three (starting from the zeroth) iterated integrals of a signature correspond to the 2D bitmap (1 feature map), direction (2 feature maps), and curvature of the pen trajectory (4 feature maps), respectively. Considering the complexity of online HCCR, we use the first three orders of the signature (7 feature maps) as input features because the contribution of integrals beyond the third order is negligible. Consequently, the path signature provides 7 feature maps from the domain knowledge layer to the succeeding convolutional layers.

5.5. 8-directional feature maps

Owing to their excellent ability to express stroke directions, 8-directional features (8Dir) [9] are widely used in HCCR. This technique extracts features from online trajectory points using their projections on 8 2D directions, and accordingly, 8 pattern images are generated and employed as feature maps in the domain knowledge layer. The number of directions can be flexibly simplified into 4 or detailed into 16 or more, but 8 directions are usually employed to strike a balance between complexity and precision.

6. Experiments

6.1. Experimental database

We used the CASIA-OLHWDB 1.0 (DB 1.0) and CASIA-OLHWDB 1.1 (DB 1.1) [41] databases, which have been set up by the Institute of Automation, Chinese Academy of Sciences. DB 1.0 contains 3740 Chinese characters in the GB2312-80 standard level-1 set (GB1) obtained from 420 writers (336×3740 samples for training, 84×3740 for testing), whereas DB 1.1 contains 3755 classes in GB1 obtained from 300 writers (240×3755 samples for training, 60×3755 for testing).

The database for the ICDAR 2013 HCCR competition [16] comprises three datasets for isolated characters (CASIA-OLHWDB 1.0∼1.2). All the data were annotated at the character level. The test dataset, which was published after the competition, was obtained from 60 writers who were not considered in DB 1.0∼DB

1.2. Note that in our experiment, we only use a combination of DB 1.0 and DB 1.1 to evaluate the ICDAR 2013 HCCR competition dataset, because the classes of DB 1.2 are outliers of the 3755 classes in GB1.

6.2. Network configurations of our DCNN

Most of the hyper-parameters in our network configurations are fixed to simplify the experimental comparison and determined based on previous works [15,73]. We use the same single DCNN structure for comparison in all experiments as that presented in Section 3, i.e., 96 × 96Input-M × 96 × 96-80C3-MP2-160C2-MP2-240C2-MP2-320C2-MP2-400C2-MP2-480FC-512FC-3755Output, with only different M to represent the different types of domain knowledge we used. A random mix of affine transformations (scaling, rotations, and translations) [15] is adopted as the basic method for data argument during the training stage. The training mini-batch size is set as 96, and the dropout [67] rates for the last four weighting layers are set as 0.05, 0.1, 0.3, and 0.2, respectively. The learning rate updates as follow:

$$\alpha(t) = \alpha(0) \cdot \exp(-\lambda t), \quad \text{where } \alpha(0) = 0.002,\ \lambda = 5 \times 10^{-6}. \tag{10}$$

The network stops when the accuracy on the training set ceases improving. We performed our experiments on a PC with Intel 3.4 Hz i7 CPU, 8 G RAM and a GTX980 GPU.

Our baseline CNN was trained with data that was directly rendered from the online handwritten Chinese characters as offline bitmaps, without using any domain-specific knowledge, as in the case of [13].

6.3. Investigation of the forgetting curve of DCNN

To verify the forgetting curve of DCNN, we used the baseline network as the experimental subject. At first, the network was pre-trained by an epoch of overall training set in CASIA-OLHWDB 1.1 to form a basic memory of knowledge. Then, 30 random classes out of the total 3755 classes were selected as the learning materials, while samples of the rest 3725 classes were regarded as random noises being reassigned another one of the labels from 3725 classes to avoid interfering with the 30 classes. After the network was trained by an epoch using all samples of the 30 classes, it was then trained with the random noises. We tested the performance of the training data of the 30 classes every mini-batch afterward and plotted the results together against the number of mini-batch iterations to get the forgetting curve. Repeatedly, after 10 times using different classes as materials, an average forgetting curve was obtained in Fig. 4. It is notable that this forgetting curve of DCNN reveals the exponential loss of information similar to that of human brain, so it supports the idea in Section 4.1 that DCNN can "forget" samples as the mini-batches passed. Thus, it is reasonable to design the *DropSample* learning method for DCNN that mimics humans.

6.4. Investigation of different online features against baseline method for HCCR

Inspired by the promising result reported in [15], we compare the iterated-integrals signature features in different orders (denoted by Sign.x) with the traditional widely used 8-directional feature [9]. By incorporating the popular online features with the Bitmap, the results are improved significantly, as shown in Fig. 5. Among them, the second-order truncated signature feature (containing 7 feature maps and denoted by Sign.2) produces better results than the first-order truncated version (3 feature maps and denoted by Sign.1) and the 8-direction feature (8 feature maps and denoted by 8Dir), and shows almost the same performance as the third-order truncated signature feature (denoted by Sign.3) which contains many more feature maps (as many as 15). As most of the domain-specific knowledge used in this paper is extracted on the basis of online information from a handwritten sample, the domain knowledge in the following experiments is extracted and applied after incorporation with the Sign.2 online feature maps, for the sake of its higher performance and reduced storage.

6.5. Investigation of the effectiveness of domain-specific knowledge

We designed nine CNNs (denoted by $A\sim I$ in Table 2) to intensively evaluate the performance achieved by the addition of different kinds of domain-specific knowledge to the baseline network. The experimental results for DB 1.1 are summarized in Table 2. From the results, the following interesting observations and conclusions can be made:

(1) The DCNNs with path signature features clearly enhance the Bitmap by adding online information. Network C using Sign.2 is better than network B using Sign.1, and is also cost-effective compared to network D using Sign.3 in terms of the number of feature maps. Similarly, all other networks with different domain knowledge outperform the baseline CNN by clear margins.

(2) Network E with DT shows a slightly better result than network C with Sign.2, while the improvement is more obvious in our preliminary small-scale experiments. Network F with LDPI [2] as the NLN method underperforms network C, although it produces better results than the other approaches in our preliminary experiments with small categories of training samples. As it was found that additional DT can somewhat extend the coverage of the possible handwriting styles, especially when data is insufficient, we keep the DT domain knowledge for further use by the ensemble model.

(3) Network G with 8-directional features produces better results than the baseline and network C, indicating that the additional features offer extra statistical information on stroke direction to enhance the performance of the baseline CNN.

(4) Network H with imaginary strokes embedded in an online character shows an obvious improvement, indicating that the imaginary stroke technique is useful.

(5) By integrating all the domain knowledge (referred to as Fusion) except NLN to network I, a significant improvement is

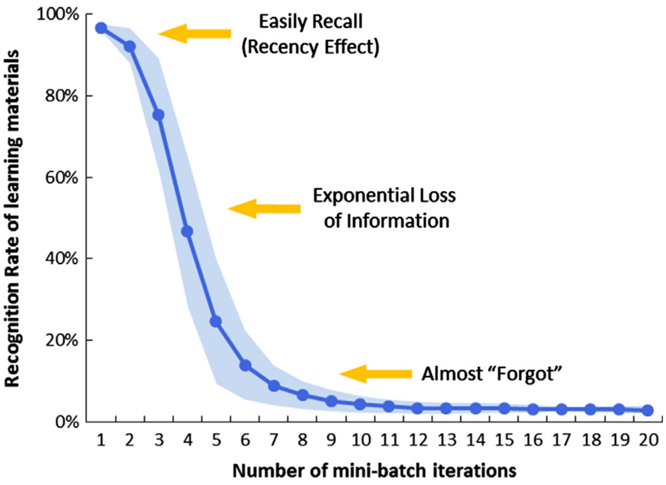

Fig. 4. The forgetting curve of DCNN. The light blue area shows the error bands of the average curve. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

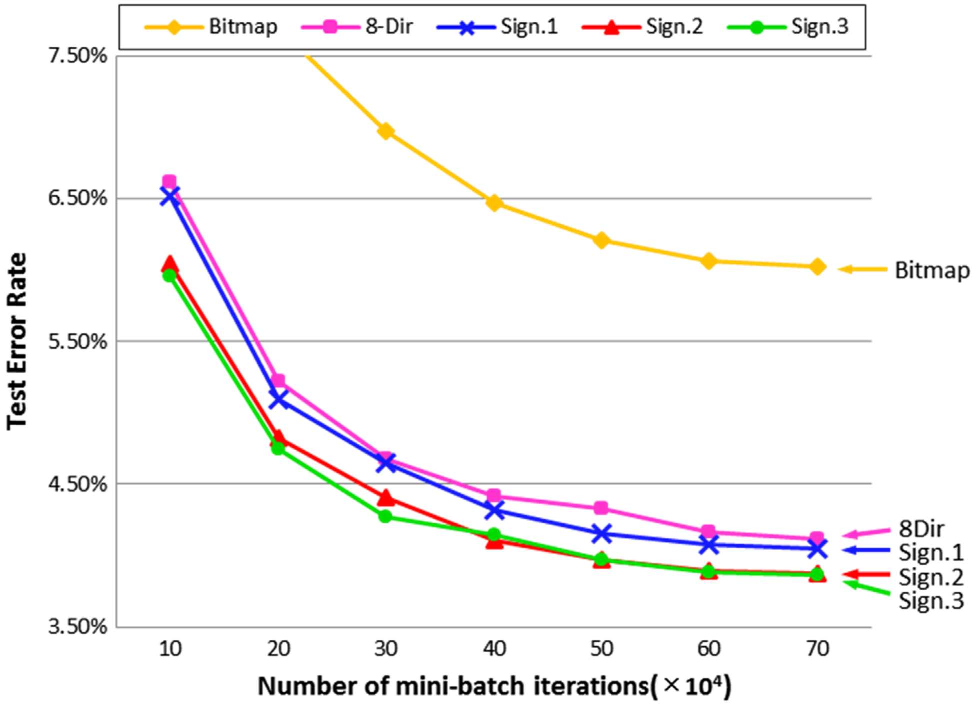

Fig. 5. Comparison of different features.

Table 2
Recognition rates (%) of different domain-specific methods on CASIA-OLHWDB 1.1.

Network	Domain-specific methods	Recognition rate (%)
A	**Baseline**:Bitmap (no domain knowledge)	93.99
B	Bitmap+Sign.1	95.95
C	Bitmap+Sign.2	96.12
D	Bitmap+Sign.3	96.12
E	Bitmap+Sign.2+DT	96.13
F	Bitmap+Sign.2+NLN	95.81
G	Bitmap+Sign.2+8Dir	96.22
H	Bitmap+Sign.2+IS	96.33
I	**Fusion**: Bitmap+Sign.2+DT+8Dir+IS	**96.39**

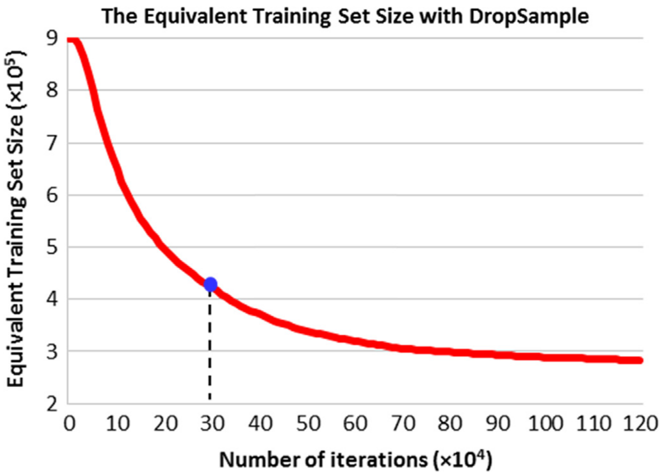

Fig. 6. Illustration of the equivalent training set size when training with *DropSample*. The initial training set size is 898,524 on CASIA-OLHWDB 1.1. Heavily noisy samples are gradually eliminated after 300,000 mini-batch iterations in our implementation.

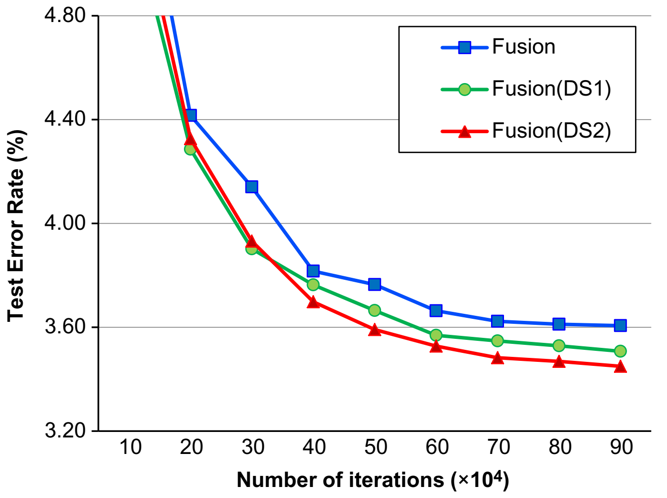

Fig. 7. Performance of the two quota-updating functions implementing *DropSample* in terms of error rate.

Table 3
Recognition rates (%) for training without or with *DropSample* on CASIA-OLHWDB 1.1.

Network	Domain-specific methods	Without *DropSample*	With *DropSample*
A	**Baseline**: Bitmap	93.99	94.26
B	Bitmap+Sign.1	95.95	96.08
C	Bitmap+Sign.2	96.12	96.30
D	Bitmap+Sign.3	96.12	96.31
E	Bitmap+Sign.2+DT	96.13	96.29
F	Bitmap+Sign.2+NLN	95.81	96.08
G	Bitmap+Sign.2+8Dir	96.22	96.36
H	Bitmap+Sign.2+IS	96.33	96.45
I	**Fusion**: Bitmap+Sign.2+DT+8Dir+IS	96.39	96.57

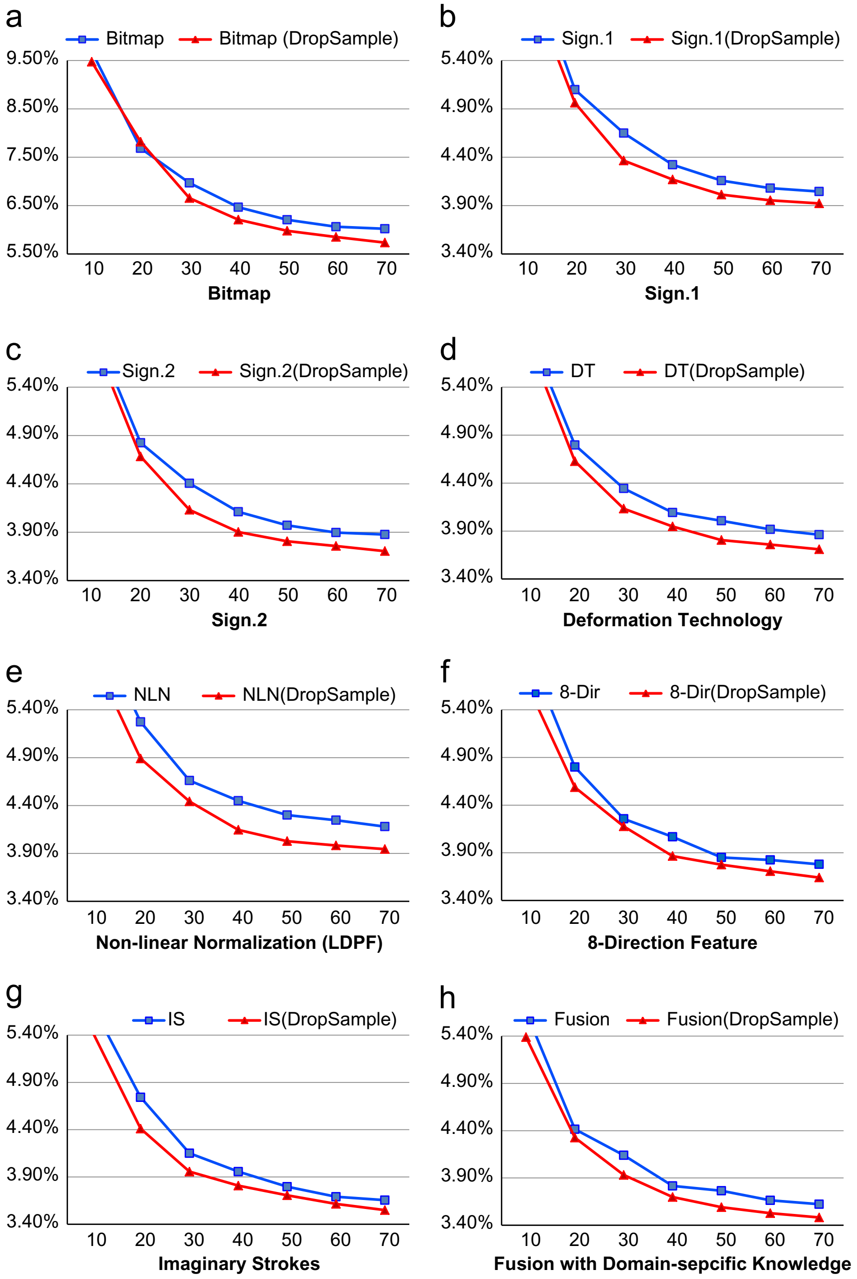

Fig. 8. Performance of different types of domain knowledge used for training with and without *DropSample* method in terms of test error rate. The *x*-axis represents the number ($\times 10^4$) of mini-batches for training.

achieved, which indicates that domain-specific knowledge is very useful for improving DCNN for HCCR.

6.6. *Investigation of DropSample training method*

First, we illustrate the number of samples dropped during the training process. The quotas are initialized to 1 so that we can obtain the equivalent training set size at each mini-batch, as shown in Fig. 6. It can be seen that the curve initially declines rapidly, and only a third of the total samples remain for further learning after a certain number of iterations (700,000 mini-batch iterations in this case); this intuitively indicates during the training process that training with *DropSample* is increasingly efficient.

Then, we conducted experiments to compare the enhancement effects of the two quota-updating methods denoted in (Eqs. (8) and 9) (denoted by DS1 and DS2, respectively). Our experiments were deliberately evaluated over the fusion network which has already achieved high performance. The results are shown in Fig. 7. It can be seen that even though the fusion network has already achieved promising results compared to the baseline, both *DropSample* methods improve the performance. The multi-level hierarchy solution (DS2) shows better results than the exponential decay solution (DS1). We further applied the *DropSample* training method (DS2) to nine networks with different kind of combinations of domain knowledge. The results are shown in Table 3 and Fig. 8. It can be seen that by applying the DropSample training method, the recognition accuracies of all the nine models were all improved.

Third, we evaluated training efficiency without and with using *DropSample*. Fig. 9 illustrates the actual training time required when both methods achieve the same recognition rates, which is the highest accuracy achieved by methods without using *DropSample*. It can be seen that the proposed *DropSample* training methods achieve

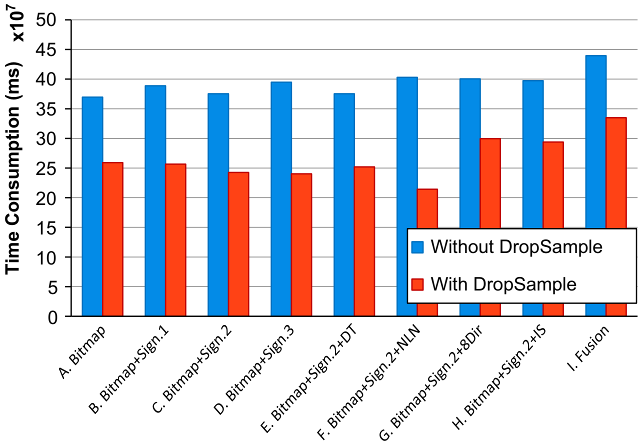

Fig. 9. Comparison of the training efficiency without and with *DropSample*, in terms of the time consumption involved for different networks. The time consumption is evaluated when both methods achieve the same recognition rates.

Table 4
Average time involved for each character at training stage without and with using *DropSample*.

Average time (ms) per character	Network								
	A	***B***	***C***	***D***	***E***	***F***	***G***	***H***	***I***
Forward propagation (T_F)	2.37	2.34	2.43	2.57	2.40	2.43	2.53	2.67	2.92
Back propagation (T_B)	2.21	2.35	2.36	2.33	2.39	2.36	2.37	2.33	2.53
Without *DropSample* ($T=T_F+T_B$)	4.58	4.69	4.79	4.90	4.79	4.79	4.90	5.00	5.45
With *DropSample* ($T_{DS}=T_Q+T$)	4.69	4.75	4.90	4.97	4.86	4.90	5.00	5.10	5.59
Quota-updating time ($T_Q=T_{DS}-T$)	0.11	0.06	0.11	0.07	0.07	0.11	0.10	0.10	0.14

the accuracies in much less time than the methods without *DropSample*, saving 23.26%~45.64% time consumptions. Moreover, we provided the detailed training time involved for each character for further comparison as shown in Table 4. We can see that even though extra time T_Q occurs in quota-updating stage, it is ignorable compared to the time for forward (T_F) and backward (T_B) propagation, because the quotas are updated in mini-batch mode instead of batch mode. *DropSample* is an efficient training method for two reasons. According to the reduction curve of training set size in Fig. 6, when using *DropSample*, the equivalent size of training epoch decreases as the iterations passed accounting for increasingly less time training over an equivalent epoch. Another reason is that the softmax output we used to update each quota is the one before back propagation rather than after it. Actually, we conducted experiments on network *I* and found that *DropSample* using either types of error results in almost the same performance, but training with error after back propagation requires much extra time for one more forward pass. Thus, the outputted error from the forward pass before back propagation is adopted in our implementation. In summary, the quantitative analysis above validates the high training efficiency of the *DropSample* method.

Finally, to quantitatively analyze the robustness of noise without and with *DropSample*, we conducted experiments on the baseline network by adding different proportions of noisy samples to the training set. Without loss of generality, we simply generated some mislabeled samples as the noisy samples in our experiments. As shown in Fig. 10, with the proportions of noisy data increasing, the test recognition rates of both methods decrease. However, as the *DropSample* method can gradually and automatically reduce the frequency of selecting mislabeled samples, it was less affected by the noise and achieves better and more robust performance. Furthermore, we evaluated the training time consumption of different networks to investigate the correlation between the proportion of noise and the training efficiency. Fig. 11 illustrates the actual training time required when both methods achieve the same recognition rates against of different proportions of noisy training samples. We can see that for different proportions of noise in the training set, the time consumptions without *DropSample* keep similar high level, while when training with the *DropSample*,

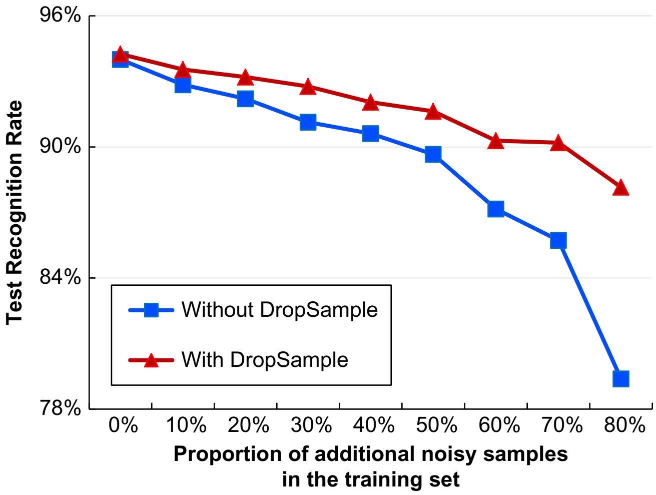

Fig. 10. Comparison of the noise robustness with and without using *DropSample* on the baseline network respectively, in terms of test recognition rate when the networks have converged.

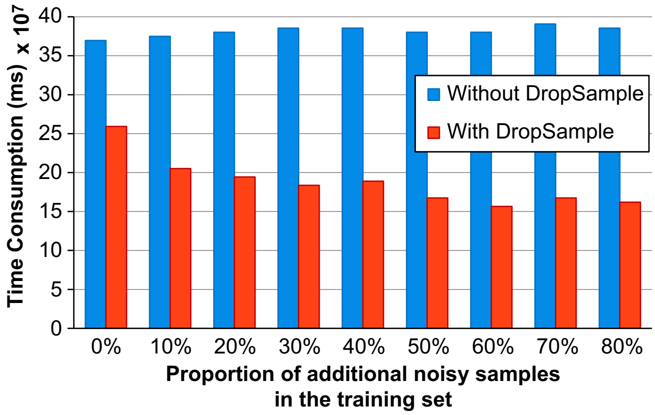

Fig. 11. Correlation between the training efficiency and the proportions of noisy training samples.

Table 5
Recognition rates (%) of the single network and the ensemble on the three CASIA-OLHW datasets.

Database	Single Network			Ensemble (Nets A~I)		
	Baseline (Bitmap)	Fusion	Training with *DropSample*	Max-pooling	Voting	Average
CASIA-OLHWDB 1.0	94.32	96.71	**97.01**	97.28	97.26	**97.33**
CASIA-OLHWDB 1.1	93.99	96.39	**96.57**	96.97	96.99	**97.06**
ICDAR 2013 competition DB	94.52	96.93	**97.23**	97.40	97.43	**97.51**

Table 6
Comparison of different methods on the three CASIA-OLHW datasets.

Database	**Published state-of-the-art performance** (%)				
	DLQDF [37]	MCDNN [13]	DeepCNet [15]	Our Single DCNN	Our ensemble
CASIA-OLHWDB 1.0	95.28	94.39	N/A	97.01	**97.33**
CASIA-OLHWDB 1.1	94.85	N/A	96.42	96.57	**97.06**
ICDAR 2013 competition DB	N/A	N/A	97.39[a]	97.23	**97.51**

[a] The result of the winner of the ICDAR 2013 HCCR competition. Note that the number of parameters of DeepCNet (5.9 million) is much larger than that of our single DCNN (3.8 million).

Table 7
Performance of large-scale unconstrained handwritten character recognition on seven datasets.

Test Set	DB 1.0 [41]	DB 1.1 [41]	DB 1.2 [41]	SCUT-COUCH [68]	HKU [6]	In-house dataset	863 [69]	Total
Number of samples	143,600	98,235	79,154	161,166	120,863	184,089	40,578	827,685
Recognition rate (%)	96.70	96.05	97.24	98.62	97.73	98.42	99.72	97.74

a lot of time computation are saved, particularly as the noises increased, accounting for improving the training efficiency.

6.7. *Evaluation of ensemble of different types of domain knowledge embedded DCNNs for HCCR*

Given a set of DCNN models with different types of domain knowledge and training with and without *DropSample*, it is useful to combine the different types of domain knowledge to achieve better performance. Our ensembling method involves averaging the softmax output of the corresponding DCNN models. As shown in Table 5, we analyzed alternative ensembling methods, but they lead to inferior performance than simple averaging. The ensemble boosts the performance and achieves a high recognition rate of 97.33% on DB 1.0 and 97.06% on DB 1.1, with relative error reduction rates of 53% and 51%, respectively, compared to the baseline method. As shown in Table 6, our final result on DB 1.1 has a test error rate of 2.94%, which is significantly lower than that of the state-of-the-art DLQDF (5.15%) [37] and DeepCNet (3.58%) [15] approaches.

Additional experiments were conducted on the ICDAR 2013 HCCR competition dataset. As multiple classifiers vary among network architectures, configurations, and embedding techniques, their ensemble achieves a recognition rate of 97.51%, which is significantly better than that of the DeepCNet method (96.65%) with relative error reduction of 26%, and is also better than the winner of the ICDAR 2013 HCCR competition (97.39%).

6.8. *Evaluation of large-scale online HCCR*

To show that the DCNN can be used to handle a very large-scale practical HCCR problem that involves a much greater number of classes (>3755), we extended the DCNN architecture to recognize as many as 10,081 classes of a mixture of handwritten Chinese characters, English letters, numerals, and symbols. The 10,081 classes of characters include 6763 simplified Chinese characters in the GB2312-80 standard, 5401 traditional Chinese characters in the Big5 standard, 52 English letters (26 upper-case and 26 lower-case letters), 10 standard numerals (0–9), 165 symbols, and 937 additional rarely used Chinese characters. Note that 3247 characters are common to both GB2312-80 and Big5, thus, there are 10,081 different classes of characters in all. The DCNN architecture we used is 96×96Input-$M\times96\times96$-100C3-MP2-200C2-MP2-300C2-MP2-400C2-MP2-500C2-MP2-600C2-1024FC-10081Output, which is similar to that presented in Section 3, but with more convolutional kernels. To evaluate this network, we use a combination of seven datasets as shown in Table 7. We list the detailed number of testing samples which are randomly selected from each dataset. The rest of the samples for each class are partitioned into training set and validation set with a ratio of 5 to 1. Since different classes may contain very different number of samples, we thus use data augmentation methods in [47,48] to ensure that all the classes have the same quantity of samples which are exactly 1,000 per class for training and 200 per class for validation. Although it took us approximately two weeks to train and optimize this DCNN system[1], the classification speed is quite fast. It takes only 12ms in average to recognize a handwritten Chinese character using a web server with 2.5 GHz CPU and 8 G memory without GPU. The results of recognition accuracies are summarized in Table 7. It can be seen that in spite of a very large number of classes (10,081), the DCNN model achieves a very promising recognition rate of 97.74% on average for a total of seven datasets. This result reflects the robust and excellent classification ability of the proposed DCNN with *DropSample* and domain knowledge enhancement.

6.9. *Discussions*

There are several noteworthy points of *DropSample*, as follows:

(a) *DropSample* can gradually change the distribution of the training set and focus on confusing samples to achieve better performance. However, large change of data distribution may lead to unbalance of training samples and inferior performance in the test stage. Fortunately, *DropSample* does not actually drop samples from the training set; instead, it provides adaptive quotas for samples to dynamically adjust their opportunity for selection as training data. In other words, every sample has the chance to be revisited and makes contribution in the overall training process.

(b) *DropSample* is designed for the datasets with a large scale and complex conditions that include various types of samples. It is more applicable for the networks which already have a fairly well performance to achieve better results efficiently training with *DropSample*. However, it is unclear if the *DropSample* can deal with very tough problems where the networks can not ensure a fair judgment of noise data. This problem is worth for further investigation.

[1] A real-time web demo of this DCNN online handwritten character recognition system is available at http://www.deephcr.net/.

(c) As the *DropSample* technique is a training method that is independent of the tasks and the network architectures, this idea shows flexibility and can be extended to other tasks and other deep models such as network in network [26], DBN [65], and stacked auto-encoder [66]. However, the functions and parameters of quota-updating in (Eqs. (8) and 9) are designed empirically in our application, a systematical way to design such functions is another issue worth for future studying.

7. Conclusion

This paper proposed *DropSample*, a new training method for DCNN, through the efficient use of training samples. A DCNN that is trained with the *DropSample* technique focuses on confusing samples and selectively ignores well-recognized samples, while effectively avoiding interference due to noisy samples. We showed that most domain-knowledge-based processing methods in the field of HCCR can enhance DCNN via suitable representation and flexible incorporation. The recognition rates of our DCNN trained with *DropSample* significantly exceeded those of state-of-the-art methods on three publicly available datasets, namely, CASIA-OLHWDB 1.0, CASIA-OLHWDB 1.1, and the ICDAR 2013 HCCR competition dataset. Furthermore, the proposed DCNN was extended to very large-scale handwritten character recognition involving 10,081 classes of characters, with millions of training data, and a promising average recognition rate of 97.737% was achieved.

Although the *DropSample* method proposed herein has been designed for DCNN models to address the large scale HCCR problem, we expect that it will also serve as a general learning technique that can be extended to other pattern recognition tasks such as image recognition, as well as other deep learning models such as deep belief networks [65] and deep recurrent neural networks [70]. These potential applications merit further investigation.

Conflict of interest

None declared.

Acknowledgement

This research is supported in part by NSFC (Grant No.: 61472144), GDSTP (Grant No.: 2013B010202004, 2014A010103012, 2015B010101004, 2015B010130003, 2015B010131004), GDUPS (2011). The authors thank all reviewers for their valuable comments on improving the quality of this paper.

References

[1] S. Leitner, So lernt man lernen: Der Weg zum Erfolg (Learning to learn: The road to success), Herder, Freiburg, 1972.
[2] C.L. Liu, K. Marukawa, Pseudo two-dimensional shape normalization methods for handwritten Chinese character recognition, Pattern Recognit. 38 (12) (2005) 2242–2255.
[3] M. Okamoto, A. Nakamura, K. Yamamoto, On-line handwriting character recognition method with directional features and directional change features, Proc. 4th Int'l Conf. Doc. Anal. Recognit. 2 (1997) 926–930.
[4] M. Okamoto, A. Nakamura, K. Yamamoto, Direction-change features of imaginary strokes for on-line handwriting character recognition, Proc. 10th Int'l Conf. Pattern Recognit. (1998) 1747–1751.
[5] M. Okamoto, K. Yamamoto, On-line handwriting character recognition using direction change features that consider imaginary strokes, Pattern Recognit. 32 (7) (1999) 1115–1128.
[6] Z.L. Bai, Q. Huo, A study on the use of 8-directional features for online handwritten Chinese character recognition, Proc. 8th Int'l Conf. Doc. Anal. Recognit. (2005) 262–266.
[7] K. Ding, G. Deng, L.W. Jin, An investigation of imaginary stroke technique for cursive online handwriting Chinese character recognition, Proc. 10th Int'l Conf. Doc. Anal. Recognit. (2009) 531–535.
[8] C.L. Liu, R. Mine, M. Koga, Building compact classifier for large character set recognition using discriminative feature extraction, Proc. 8th Int'l Conf. Doc. Anal. Recognit. (2005) 846–850.
[9] F. Kimura, K. Takashina, S. Tsuruoka, Y. Miyake, Modified quadratic discriminant functions and the application to Chinese character recognition, IEEE Trans. Pattern Anal. Mach. Intell. 1 (1987) 149–153.
[10] T. Long, L.W. Jin, Building compact MQDF classifier for large character set recognition by subspace distribution sharing, Pattern Recognit. 41 (9) (2008) 2916–2925.
[11] X. Gao, L.W. Jin, J.X. Yin, J.C. Huang., A new SVM-based handwritten Chinese character recognition method, Acta Electron. Sin. 30 (5) (2002) 651–654.
[12] H.J. Kim, K.H. Kim, S.K. Kim, J.K. Lee, On-line recognition of handwritten Chinese characters based on hidden Markov models, Pattern Recognit. 30 (9) (1997) 1489–1500.
[13] D.C. Cireşan, U. Meier, J. Schmidhuber, . Multi-column deep neural networks for image classification, Proc. 2012 IEEE Conf. Comput. Vision. Pattern Recognit. (2012) 3642–3649.
[14] D.C. Cireşan, J. Schmidhuber, Multi-column deep neural networks for offline handwritten Chinese character classification, arXiv Prepr. arXiv 1309 (2013) 0261.
[15] B. Graham, Sparse arrays of signatures for online character recognition, arXiv Prepr. arXiv 1308 (2013) 0371.
[16] F. Yin, Q.F. Wang, X.Y. Zhang, et al., ICDAR 2013 Chinese handwriting recognition competition, Proc. 12th Int'l Conf. Doc. Anal. Recognit. (2013) 1464–1470.
[17] C. Wu, W. Fan, Y. He, et al., Handwritten character recognition by alternately trained relaxation convolutional neural network, ICFHR (2014).
[18] G.E. Hinton, N. Srivastava, A. Krizhevsky, et al., Improving neural networks by preventing co-adaptation of feature detectors, arXiv Prepr. arXiv 1207 (2012) 0580.
[19] L. Wan, M. Zeiler, S. Zhang, et al., Regularization of neural networks using dropconnect, Proc. 30th Int.'l Conf. Mach. Learn. (2013) 1058–1066.
[20] Y. Bengio, P. Lamblin, D. Popovici, et al., Greedy layer-wise training of deep networks, Adv. Neural Inf. Process. Syst. 19 (2007) 153.
[21] J.D. Owens, M. Houston, D. Luebke, et al., GPU computing, Proc. IEEE 96 (5) (2008) 879–899.
[22] Y. LeCun, B. Boser, J.S. Denker, et al., Handwritten digit recogntion with a back-propagation network, Adv. Neural Inf. Process. Syst. (1990).
[23] Y. LeCun, L. Bottou, Y. Bengio, et al., Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[24] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. (2012) 1097–1105.
[25] C. Szegedy, W. Liu, Y. Jia, et al., Going deeper with convolutions, arXiv Prepr. arXiv 1409 (2014) 4842.
[26] M. Lin, Q. Chen, S.C. Yan. Network in network. arXiv preprint arXiv:1312.4400, 2013.
[27] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[28] Y. Taigman, M. Yang, M.A. Ranzato, et al., Deepface: Closing the gap to human-level performance in face verification, Proc. 2014 IEEE Conf. Comput. Vision. Pattern Recognit. (2014) 1701–1708.
[29] Y. Sun, Y. Chen, X. Wang, X. Tang, Deep learning face representation by joint identification-verification, Adv. Neural Inf. Process. Syst. (2014) 1988–1996.
[30] D.C. Cireşan, U. Meier, L.M. Gambardella, et al., Convolutional neural network committees for handwritten character classification, Proc. 11th Int'l Conf. Doc. Anal. Recognit. (2011) 1135–1139.
[31] A. Toshev, C. Szegedy, Deeppose: Human pose estimation via deep neural networks, arXiv Prepr. arXiv 1312 (2013) 4659.
[32] T. Wang, D.J. Wu, A. Coates, A.Y. Ng, . End-to-end text recognition with convolutional neural networks, Proc. 21st Int.'l Conf. Pattern Recognit. (2012) 3304–3308.
[33] A. Coates, B. Carpenter, C. Case, et al., Text detection and character recognition in scene images with unsuperivised feature learning, Proc. 11th Int.'l Conf. Doc. Anal. Recognit. (2011) 440–445.
[34] M. Jaderberg, A. Vedaldi, A. Zisserman, Deep features for text spotting, ECCV (2014) 512–528.
[35] A. Bissacco, M. Cummins, Y. Netzer, H. Neven, Photoocr: Reading text in uncontrolled conditions, Proc. 2013 IEEE Int.'l Conf. Comput. Vision. (2013) 785–792.
[36] M. Jaderberg, K. Simonyan, A. Vedaldi, A. Zisserman, Synthetic data and artificial neural networks for natural scene text recognition, arXiv Prepr. arXiv 1406 (2014) 2227.
[37] C.L. Liu, F. Yin, D.H. Wang, et al., Online and offline handwritten Chinese character recognition: Benchmarking on new databases, Pattern Recognit. 46 (1) (2013) 155–162.
[38] L. Prechelt, Early stopping-but when? Neural Netw.: Tricks trade (1998) 55–69.
[39] A.D. Baddeley, Human Memory: Theory and Practice, Psychology Press, 1997.
[40] G. Montavon, G. Orr, K. Müller, Neural Networks: Tricks of the Trade. Number LNCS 7700 in Lecture Notes in Computer Science Series, Springer, 2012.
[41] C.L. Liu, F. Yin, D.H. Wang, et al., CASIA online and offline Chinese handwriting databases, Proc. 11th Int'l Conf. Doc. Anal. Recognit. (2011) 37–41.

[42] H. Fujisawa, Forty years of research in character and document recognition—An industrial perspective, Pattern Recognit. 41 (8) (2008) 2435–2446.
[43] C.L. Liu, K. Marukawa, Global shape normalization for handwritten Chinese character recognition: A new method, Proc. 9th Int'l Workshop Front. Handwrit. Recognit. (2004) 300–305.
[44] T. Horiuchi, R. Haruki, H. Yamada, K. Yamamoto, Two-dimensional extension of nonlinear normalization method using line density for character recognition, Proc. 4th IEEE Int.'l Conf. Doc. Anal. Recognit. 2 (1997) 511–514.
[45] H. Miyao, M. Maruyama, Virtual example synthesis based on PCA for off-line handwritten character recognition, Doc. Anal. Syst. VII (2006) 96–105.
[46] G. Chen, H.G. Zhang, J. Guo, Learning pattern generation for handwritten Chinese character using pattern transform method with cosine function, Int.'l Conf. Mach. Learn. Cybern. (2006) 3329–3333.
[47] K.C. Leung, C.H. Leung, Recognition of handwritten Chinese characters by combining regularization, Fisher's discriminant and distorted sample generation, Proc. 10th Int'l Conf. Doc. Anal. Recognit. (2009) 1026–1030.
[48] L.W. Jin, J.C. Huang, J.X. Yin, Q.H. He, A novel deformation on transformation and its application to handwritten Chinese character shape correction, J. Image Graph. 7 (2) (2002) 170–175.
[49] A. Biem, S. Katagiri, B.H. Juang, Pattern recognition using discriminative feature extraction, IEEE Trans. Signal Process. 45 (2) (1997) 500–504.
[50] K. Fukushima, Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, Biol. Cybern. 36 (4) (1980) 193–202.
[51] P.Y. Simard, D. Steinkraus, J.C. Platt, Best practices for convolutional neural networks 958-958, Proc. 12th Int.'l Conf. Doc. Anal. Recognit. 2 (2003).
[52] I.J. Kim, X. Xie, Handwritten Hangul recognition using deep convolutional neural networks, Int. J. Doc. Anal. Recognit. (2014) 1–13.
[53] D.C. Cireşan, U. Meier, L.M. Gambardella, J. Schmidhuber, Convolutional neural network committees for handwritten character classification, Proc. 11th IEEE Int.'l Conf. Doc. Anal. Recognit. (2011) 1135–1139.
[54] S. Wshah, V. Govindaraju, Y. Cheng, H. Li, A novel lexicon reduction method for Arabic handwriting recognition, Proc. 20th IEEE Int.'l Conf. Pattern Recognit. (2010) 2865–2868.
[55] L.L. Ma J. Wu, A Tibetan component representation learning method for online handwritten Tibetan character recognition, 2010.
[56] K. Mehrotra, S. Jetley, A. Deshmukh, et al., Unconstrained handwritten Devanagari character recognition using convolutional neural networks. ACM Proc. 4th Int'l Workshop on Multilingual OCR, 2013.
[57] F. Yin, M.K. Zhou, Q.F. Wang, C.L. Liu, Style consistent perturbation for handwritten chinese character recognition, Proc. 12th Int'l Conf. Doc. Anal. Recognit. (2013) 1051–1055.
[58] B. Hambly, T. Lyons, Uniqueness for the signature of a path of bounded variation and the reduced path group, Ann. Math. 2 (171) (2010) 109–167.
[59] V. Nair, G. Hinton, Rectified linear units improve restricted boltzmann machines, Proc. 27th Int'l Conf. Mach. Learn. (2010).
[60] C. Özkan, F.S. Erbek., The comparison of activation functions for multispectral Landsat TM image classification, Photogramm. Eng. Remote. Sens. 69 (11) (2003) 1225–1234.
[61] ImageNet large scale visual recognition challenge 2013 Result. ⟨http://www.image-net.org/challenges/⟩ LSVRC/2013/results.php.
[62] Y. LeCun, L. Bottou, G.B. Orr, et al., Efficient Backprop. Book Chapter, Neural Networks: Tricks of the Trade, Springer Berlin Heidelberg (2012), p. 9–48.
[63] A. Yuan, G. Bai, L. Jiao, et al., Offline handwritten English character recognition based on convolutional neural network, Proc. 10th IAPR Int.'l Workshop Doc. Anal. Syst. (2012) 125–129.
[64] H. Ebbinghaus, Memory: A Contribution to Experimental Psychology. Teachers College, Columbia University, 1913.
[65] G.E. Hinton, L. Deng, D. Yu, et al., Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, IEEE Signal Process. Mag. 29 (6) (2012) 82–97.
[66] P. Vincent, H. Larochelle, Y. Bengio, P.A. Manzagol, . Extracting and composing robust features with denoising autoencoders, ACM. Proc. 25th Int. Conf. Mach. Learn. (2008) 1096–1103.
[67] G.E. Hinton, N. Srivastava, A. Krizhevsky, et al., Improving neural networks by preventing co-adaptation of feature detectors, arXiv Prepr. arXiv 1207 (2012) 0580.
[68] L.W. Jin, Y. Gao, G. Liu, et al., SCUT-COUCH2009: A comprehensive online unconstrained Chinese handwritting database and benchmark evaluation, Int. J. Doc. Anal. Recognit. 14 (1) (2011) 53–64.
[69] Y.L. Qian, S.X. Lin, Q. Liu, et al., Design and construction of HTRDP corpus resources for Chinese language processing and intelligent human-machine interaction, Chin. High. Technol. Lett. 15 (1) (2005) 107–110.
[70] A.Graves, A.R. Mohamed and G.E. Hinton. Speech recognition with deep recurrent neural networks. 2013 IEEE Int'l Conf. on Acoustics, Speech and Signal Processing, 2013, 6645-6649.
[71] M.J. He, S.Y. Zhang, H.Y. Mao, L.W. Jin, Recognition confidence analysis of handwritten Chinese character with CNN, Proc. 13th IEEE Int.'l Conf. Doc. Anal. Recognit. (2015) 61–65.
[72] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Journal. Comput. Syst. Sci. 55 (1) (1997) 119–139.
[73] W.X. Yang, L.W. Jin, Z.C. Xie, Z.Y. Feng, Improved deep convolutional neural network for online handwritten Chinese character recognition using domain-specific knowledge, Proc. 13th IEEE Int.'l Conf. Doc. Anal. Recognit. (2015) 551–555.

Weixin Yang received the B.S. degree from the College of Electronic and Information Engineering at the South China University of Technology, Guangzhou, China in 2013. He is currently pursuing the Ph.D. degree in information and communication engineering at the South China University of Technology, Guangzhou, China. His current research interests include machine learning, handwriting analysis and recognition, computer vision.

Lianwen Jin received the B.S. degree from the University of Science and Technology of China, Anhui, China, and the Ph.D. degree from the South China University of Technology, Guangzhou, China, in 1991 and 1996, respectively. He is currently a Professor with the School of Electronic and Information Engineering, South China University of Technology. He is the author of more than 100 scientific papers. Dr. Jin is a member of the China Image and Graphics Society and the Cloud Computing Experts Committee of the China Institute of Communications. He was a recipient of the award of New Century Excellent Talent Program of MOE in 2006 and the Guangdong Pearl River Distinguished Professor Award in 2011. He served as a program committee member for a number of international conferences, including ICFHR2008- ICFHR2016, ICDAR2009-ICDAR2015, ICPR2010-ICPR2016, ICME2014-ICME2016, ICIP2014, ICIP2016, etc. His current research interests include handwriting analysis and recognition, pattern recognition and machine learning, image processing and intelligent systems.

Dacheng Tao is currently a Professor of computer science at the Centre for Quantum Computation and Intelligent Systems and the Faculty of Engineering and Information Technology in the University of Technology, Sydney, Australia. He mainly applies statistics and mathematics for data analysis problems in data mining, computer vision, machine learning, multimedia, and video surveillance. He has authored and co-authored more than 100 scientific articles at top venues including IEEE T-PAMI, T-NNLS, PR, T-IP, NIPS, ICML, AISTATS, ICDM, CVPR, ICCV, ECCV; ACM T-KDD, KDD and Multimedia, with the best theory/algorithm paper runner up award in IEEE ICDM '07.

Zecheng Xie received the B.S. degree from the College of Electronic and Information Engineering at the South China University of Technology, Guangzhou, China. He is currently pursuing the Ph.D. degree in information and communication engineering at the South China University of Technology, Guangzhou, China. His current research interests include machine learning, handwriting analysis and recognition, computer vision.

Ziyong Feng received the B.S. degree from the College of Engineering at South China Agricultural University, Guangzhou, China. He is currently pursuing the Ph.D. degree in information and communication engineering at the South China University of Technology, Guangzhou, China. His current research interests include machine learning, computer vision.