Pattern Recognition 77 (2018) 395–411

Contents lists available at ScienceDirect

Pattern Recognition

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

DropRegion training of inception font network for high-performance Chinese font recognition

Shuangping Huang, Zhuoyao Zhong, Lianwen Jin*, Shuye Zhang, Haobin Wang

School of Electronic and Information Engineering, South China University of Technology, Guangzhou 510641, China

ARTICLE INFO

Article history:
Received 15 March 2017
Revised 10 September 2017
Accepted 15 October 2017
Available online 17 October 2017

Keywords:
Chinese font recognition
DropRegion
Elastic meshing technique
Inception font network (IFN)
Text block-based font recognition

ABSTRACT

Chinese font recognition (CFR) has gained significant attention in recent years. However, due to the sparsity of labeled font samples and the structural complexity of Chinese characters, CFR is still a challenging task. In this paper, a DropRegion method is proposed to generatea large number of stochastic variant font samples whose local regions are selectively disrupted and an inception font network (IFN) with two additional convolutional neural network (CNN) structure elements, i.e., a cascaded cross-channel parametric pooling (CCCP) and global average pooling, is designed. Because the distribution of strokes in a font image is non-stationary, an elastic meshing technique that adaptively constructs a set of local regions with equalized information is developed. Thus, DropRegion is seamlessly embedded in the IFN, which enables end-to-end training; the proposed DropRegion-IFN can be used for high performance CFR. Experimental results have confirmed the effectiveness of our new approach for CFR.

1. Introduction

Font recognition plays an important role in document analysis and recognition of character images. It is also fundamental to graphic design, as font is a core design element of any printed or displayed text. However, font recognition has long been neglected by the vision community, despite its importance. Font recognition is an inherently difficult and error-prone task because of the huge number of available fonts, the dynamic and open-ended properties of font classes, and the very subtle and character-dependent differences among fonts [1]. Over the years, some approaches to font recognition have been proposed [2–11], but most font recognition research has been carried out on text in Western languages, and the approaches proposed have not yielded satisfactory accuracy. Because of the structure and ideographic nature of individual Chinese characters, Chinese font recognition is more challenging than recognition of most Western scripts. The limited number of previous studies [5,9–10,11,43] of Chinese font recognition have mostly approached the problem from a document analysis perspective, which makes the results highly sensitive to noise and only applicable to simple cases with strong constraints. In this study, therefore, we approached the development of a more effective automatic Chinese font recognition method from a computer vision perspective.

Recently, deep convolutional neural networks (CNNs) have achieved great success in many computer vision tasks, such as image classification [12–16,20], handwritten character recognition [22,24,31,37,44], and object detection [25,26,34,40]. These CNN methods receive raw data and automatically learn the representations needed for specific tasks. Beyond the conventional pattern analysis pipeline, which involves feature extraction and a classifier design, a CNN is a type of end-to-end representation learning framework. Font recognition, on the other hand, can be regarded as a special form of image classification problem. Based on this analysis, we consider Chinese font recognition as a representation learning problem modeled by CNNs. The current explosion of research on the use of CNNs for deep learning in computer vision has produced many good examples of CNN networks, including AlexNet [12], the Zeiler and Fergus model [13], VGGNet [14], GoogleNet [18] and others. However, use of these networks is not guaranteed to result in improved performance. Careful design of a CNN-based network is required for the specific task of Chinese font recognition.

An inception-type network is at the core of most state-of-the-art computer vision solutions based on CNNs [15,17]. This inception architecture has made it feasible to use CNN-based solutions in big-data scenarios to achieve higher performance without adding extra complexity. The basic inception-style building block introduces parallel structures that run convolutional layers on several different scales, plus an extra pooling layer, and concatenates their responses. In this way, the impact of structural changes on nearby components is mitigated to adapt to the diversity of data samples.

* Corresponding author.
E-mail addresses: eehsp@scut.edu.cn (S. Huang), z.zhuoyao@mail.scut.edu.cn (Z. Zhong), eelwjin@scut.edu.cn (L. Jin), potatoandleave@163.com (S. Zhang), wang.haobin@mail.scut.edu.cn (H. Wang).

https://doi.org/10.1016/j.patcog.2017.10.018
0031-3203/

Chinese characters are characterized by their elaborate structures. These structures are rendered on different scales of local patches, comprehensively reflecting font identification information. Therefore, it is reasonable to assume that inception is a suitable network structure choice for Chinese font recognition. The general design principles of inception-type networks have been presented in previous studies [15,17]. Research to date has not shown, however, that use of an inception-type network, e.g., GoogleNet, will yield significant quality gains in practical scenarios. To this end, an inception font network (IFN) was constructed in this study specifically for use in Chinese font recognition.

It has been determined empirically that a classifier can distinguish characters correctly even when they are partially obscured. This capability is similar to that of human beings, who can correctly discriminate a visual object using the surrounding context. In other words, regardless of whether a local region exists, the font category is not changed. Inspired by the above observation and analysis, we sought to improve generalization performance by introducing disrupted samples during the training phase of the IFN. In this study, we formulated this occlusion process as DropRegion. Specifically, during the training process, one image is divided into several regions. Then, one or more local regions are randomly selected and destructed by a type of noise. This process is embedded in the IFN, and the whole framework can be trained end to end by back propagation and stochastic gradient descent. It has been observed that printed characters pose a variety of spatial layouts, while the stroke information over the character image centroid is significantly more compact than at the boundary. To cater to the distribution characteristics of character structures, we developed an elastic mesh technique for programming the region division in DropRegion. Basically, an elastic mesh technique is used to divide an image elastically into several parts, ensuring that each part contains an equal amount of information.

To summarize, we present a new approach to Chinese font recognition, DropRegion-IFN, that integrates an IFN and a new model regularization method called DropRegion, which randomly removes several elastically sized regions from the characters of an original Chinese character prototype while retaining the identity information contained in it. The proposed DropRegion implements data augmentation, thus improving the generalized applicability of the CNN-based network model and preventing model overfitting. Furthermore, we introduce IFN for sufficient feature representation, which caters to the DropRegion training mode and specifically considers the Chinese font recognition task.

The remainder of this paper is organized as follows. Related research is summarized in Section 2. The proposed DropRegion-IFN method is described in Section 3. Single character-based and text block-based font recognition schemes are described in Sections 4 and 5, respectively. Experimental results are presented in Section 6. Conclusions drawn from the results of the research are presented in Section 7.

2. Related studies

Researchers typically regard font recognition as a pattern recognition task that involves feature extraction and classifier design. Various methods that emphasize feature extraction or classifier design have been proposed in the font recognition field [6-10]. For example, Cooperman [6] used local detectors to identify regions that have typographic properties (e.g., serif, boldness, and italics). Zramdini et al. [7] employed a scale-invariant feature transform algorithm to build an Arabic font recognition system. Zhu et al. [8] extracted the texture features of a text block containing several characters using a group of Gabor filters. Ding et al. [9] extracted wavelet features from a text image using a method that yielded good performance with text images of different sizes. Tao et al. [10] used multiple local binary patterns to describe the discriminative information in a text block. These studies focused on extraction of different features, which suggests that typeface feature descriptors dictate the accuracy of font recognition. Beyond these conventional feature extraction methods, the biologically inspired feature manifold scheme [47-48] provides a better alternative for the font classification task. However, some useful information may easily be lost because of the fixed handcrafted feature representation rule or manifold dimension reduction. Several attempts to improve overall classification performance have involved designing complex learning algorithms that are performed on extracted features. Zhu et al. [8] applied a weighted Euclidean distance classifier to Chinese font information. Ding et al. [9] used modified quadratic discriminant functions to enhance classification power. Slimane et al. [3] used Gaussian mixture models to build an Arabic font recognition system. Zhang et al. [11] used a support vector machine to distinguish among 25 types of Chinese fonts. All of these studies involved the usual approach of feature extraction followed by classifier design, which requires careful engineering and considerable domain knowledge and being separated into two stages completely.

Going beyond the “feature extraction plus classifier design” pipeline, CNNs have been used in end-to-end approaches to jointly learn features representation and as a classifier. CNNs have been used with great success in the field of computer vision [12–15,17–21]. However, to the best of our knowledge, there is no Chinese font recognition method based on CNNs, although one related study [1] proposed the use of CNNs for Roman alphabets. Wang et al. [1] developed the DeepFont system, which relies on hierarchical deep CNNs for domain adoption to compensate for real-to-synthetic domain gaps and improve Roman character font recognition. This system employs a basic CNN that is similar to the popular AlexNet structure for ImageNet [12]. Another system worth mentioning is the principal component 2DLSTM (2-D long short-term memory) algorithm proposed by Tao et al. [43], in which a principal component layer convolution operation is introduced to handle noisy data and take advantage of 2DLSTM to capture the contrast between a character trajectory and the background. However, this algorithm applies only to Chinese font recognition based on single characters; it is not applicable to font identification based on Chinese text blocks.

CNNs are hierarchical architectures that are stacked with multiple non-linear modules and have powerful abilities in mapping inputs to outputs. However, being neural networks, CNNs can easily become trapped in local minima when inappropriately handled. When this happens, degradation of representation learning [27] performance may occur. To improve the generalization performance or reduce overfitting, researchers have proposed several effective methods, including Dropout [28], DropConnect [29], model ensemble learning [23], and others. During the training process, Dropout randomly drops a subset of hidden neurons, while DropConnect stochastically excises a subset of connections between two hidden layers. The ensemble method is also an effective technique for reducing generalization errors by combining multiple models. Huang et al. [30] recently developed a training algorithm that drops a random subset of layers into a deep residual network [19] and achieves good performance in reducing overfitting. Bastien et al. [31] developed a powerful generator of stochastic variations for handwritten character images. Simard et al. [32] expanded the training set by adding elastically distorted samples for use in visual document analysis. In this paper, we propose a new method called DropRegion to improve generalization performance by introducing disrupted samples into the training process rather than by regularizing activations, weights, or layers. In DropRegion, some small regions are randomly dropped from the characters, and the remaining regions are recombined to form a new character. Us-

ing DropRegion, a large number of new and diverse characters can be generated from a prototype character, thereby solving the problem of scarcity of labeled font image samples. It should be noted that Gidaris et al. [34] applied multiple regions, including the original candidate box, half boxes, central regions, border regions, and contextual regions, for learning a detection-oriented model. They claimed that half boxes, which utilize each half of an object, make the features more robust to occlusions. In our approach, instead of blocking fixed parts (left, right, top, bottom) of an object, a more flexible strategy is adopted that randomly disrupts one or more small regions in each text image during the training process.

Since Google introduced the "inception module" for state-of-the-art image classification in 2014 [18], inception has been a central component of the deep convolutional neural architecture that has driven CNN development. The inception architecture has since been refined in various ways, first by the introduction of batch normalization [35] and later by additional factorization ideas [15]. Furthermore, inception architecture was combined with residual connections to develop Inception-ResNet, which won the first place in the 2016 ILSVRC classification task competition [17]. A few general principles and optimization ideas related to inception are described in these studies, but these principles do not necessarily yield performance gains in practical use. It is also impossible to apply the inception network structure proposed in the abovementioned studies to a particular application case. To this end, we explored the potential advantages of designing an inception-integrated network for Chinese fonts, with the goal of improving CFR performance based on high-quality, learned visual features.

Cascaded cross-channel parametric pooling (CCCP) was introduced by Lin et al. [36] to allow complex and learnable interactions of cross-channel information. CCCP has considerable capabilities in modeling various distributions of latent concepts, using a network-in-network (NIN) structure to achieve better local abstraction. Another important idea proposed by Lin et al. [36] is global average pooling over feature maps, which functions as the counterpart of CCCP. Global average pooling enforces correspondence between feature maps and categories, providing a structural regularizer that reduces the effect of overfitting. In this study, we developed an IFN, equipped with inception, CCCP, and global average pooling network structure components, for use in Chinese font recognition. This approach was motivated by the following three aspects of the problem: 1) Chinese language has a large set of characters, and these characters vary widely in structure and grey-level distribution. Because Chinese characters carry font discrimination information in local structures of different scales with different character construction, inception is suitable for application to CFR because inception employs different kernel sizes to capture correlation structures of different scales. 2) Diverse and overlapping strokes can be abstracted well using CCCP. 3) The proposed DropRegion yields a large number of diverse samples to compensate for the scarcity of labeled training data. To reduce overfitting caused by network complexity, global averaging pooling is employed as a counterpart to CCCP.

3. Proposed DropRegion-IFN method

3.1. Inception font network

We have designed an IFN specifically for Chinese font recognition by combining an inception module, CCCP layers, and global average pooling. The structure of the IFN is illustrated in Fig. 1.

As Fig. 1 shows, we modified the most common type of inception module by adding three branches: one 3×3 convolution followed by two stacked 2×2 convolutions, two stacked 2×2 convolutions, and two stacked 3×3 convolutions. The first and second branches have an effective receptive field of 5×5, and the third branch has an effective receptive field of 3×3. Factorizing a larger convolution into several smaller convolutions, e.g., factorizing 5×5 into two stacked 3×3 convolutions or factoring 3×3 into two stacked 2×2 convolutions, makes it possible to increase the depth and width of the network while maintaining a consistent computational budget. This application of multi-scale convolutional kernels (2×2, 3×3, and 5×5 convolutions) reflects the basic concept of the inception approach, i.e., that visual information should be processed at various scales and then aggregated so that the subsequent stage can abstract features simultaneously from different scales. After the inception module operation, several feature maps with the same sizes are obtained by means of a padding strategy and precise designs. Finally, the feature maps are concatenated using a concatenation layer. Because of these additional branches, more non-linear rectification layers can be incorporated into our model, enhancing the advantages and feasibility of the inception module. Furthermore, because of the inception module, we can extract local feature representations using more flexible convolutional kernels, which are filters of various sizes organized in a layer-by-layer form.

Given that Chinese character structure is of critical importance in font recognition, we use multiple CCCP layers, as shown in Fig. 1. Pairs of adjacent CCCP layers are stacked together, and an extra traditional convolution layer is added below each pair of CCCP layers, resulting in a micro neural network (referred to as mlpconv in paper [36]) to abstract the data within the underlying patch. Nonlinear activation functions (ReLU) are embedded between the CCCP layers, increasing the level of abstraction. This micro neural network takes the place of a traditional convolution kernel, sliding over the input, obtaining the feature maps, and feeding into the next layer. This sliding mode makes the micro network shared among all the local receptive fields. As Fig. 1 shows, a total of three micro neural networks are used in our IFN, two of which are stacked below the inception module and the third of which is stacked on top of it. The layout of the micro neural network in relation to the core inception module is intended to achieve better abstraction for features of all levels. The last micro neural network is covered by an additional layer of 1×1 convolutions to reduce the representational dimension to the number of font categories. Thus far in the process, explicit confidence maps for all of the Chinese font categories are obtained. The final step is global average pooling for each map to obtain the resulting confidence vector that corresponds to the font categories, saving a large number of parameters. As described in [36], global average pooling can be viewed as a structural regularizer that reduces overfitting. The architectural parameters tuned specifically for single character-based and text block-based Chinese font recognition are shown in Tables 1 and 2, respectively, in Section 5.

3.2. DropRegion

The idea of multiple regions—including the original candidate box, half boxes, central regions, border regions, and contextual regions—was used by Gidaris et al. [34] to train a detection-oriented model. They proved that using half boxes is an effective way to improve occlusion detection performance. In our approach, instead of blocking fixed parts (left/right/top/bottom) of an object, we adopt a more flexible scheme to randomly disrupt one or more small regions of a text image. First, an elastic meshing technique is used to divide a character or text block image into a set of small regions. Each region contains an equal amount of information. Next, a well-designed strategy guides the algorithm in disrupting the selected region. During the training phase, it is uncertain which regions are blocked because the algorithm permits blocking of any text image region in each iteration. Consequently, the training set size is effectively increased, and the diversity of

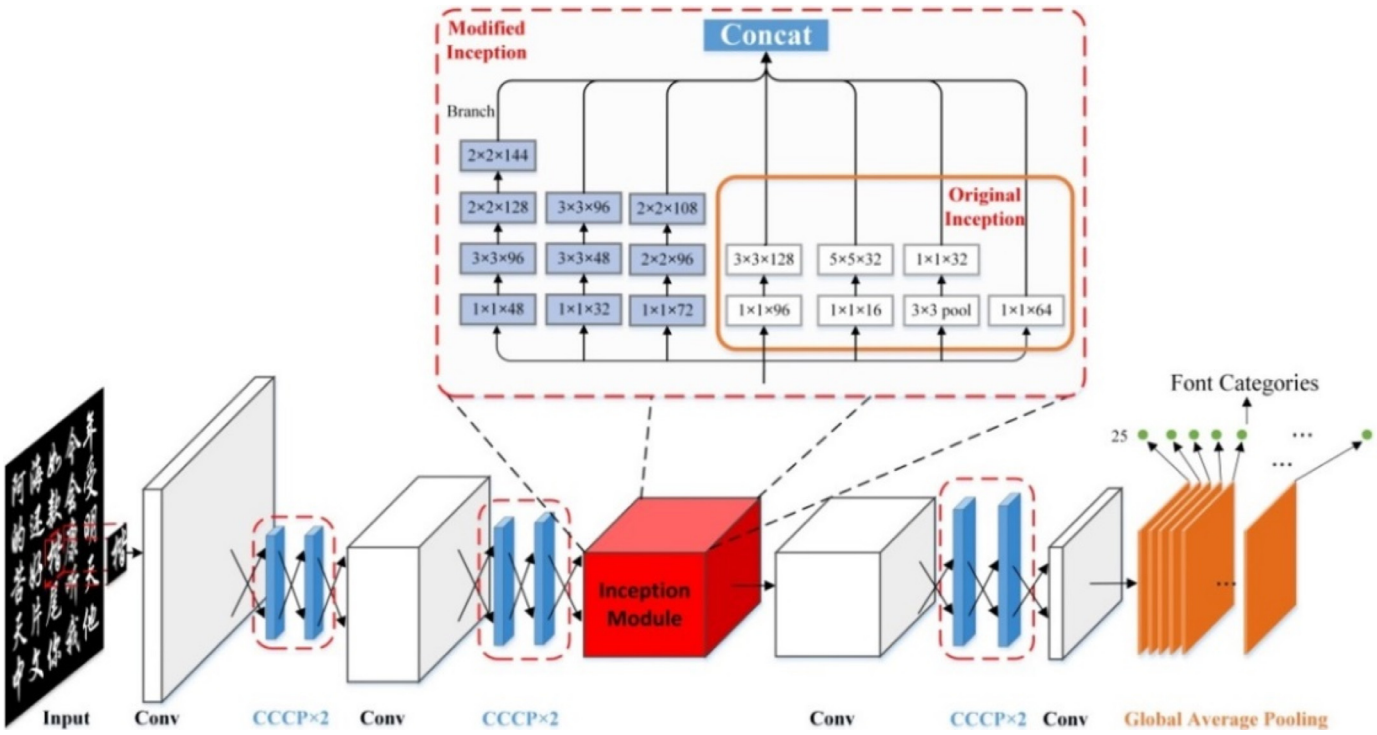


Fig. 1. Schematic illustration of IFN structure.

Table 1
Network parameters for single-character font recognition (*SingleChar-IFN*).

SingleChar-IFN		
Type	Settings	Output size
Input		$64 \times 64 \times 1$
Conv 1	$96 \times 7 \times 7$	$58 \times 58 \times 96$
CCCP 1_1	$96 \times 1 \times 1$	$58 \times 58 \times 96$
CCCP 1_2	$96 \times 1 \times 1$	$58 \times 58 \times 96$
Max - pooling	3×3, st. 2	$29 \times 29 \times 96$
Conv 2	$256 \times 7 \times 7$	$23 \times 23 \times 256$
CCCP 2_1	$256 \times 1 \times 1$	$23 \times 23 \times 256$
CCCP 2_2	$256 \times 1 \times 1$	$23 \times 23 \times 256$
max - pooling	3×3, st. 2	$11 \times 11 \times 256$
Modified Inception	detailed in Fig. 1	$11 \times 11 \times 604$
Conv 3	$512 \times 3 \times 3$, pad 1	$11 \times 11 \times 512$
CCCP 3_1	$512 \times 1 \times 1$	$11 \times 11 \times 512$
CCCP 3_2	$512 \times 1 \times 1$	$11 \times 11 \times 512$
Dropout	0.5	
Conv 4	$d \times 1 \times 1$	$11 \times 11 \times d$
Global Ave Pooling	11×11	$1 \times 1 \times d$

Table 2
Architectures of the text block network (*TextBlock-IFN*).

TextBlock-IFN		
Type	Settings	Output size
Input		$128 \times 128 \times 1$
Conv 1	$96 \times 7 \times 7$, st. 2	$61 \times 61 \times 96$
CCCP 1_1	$96 \times 1 \times 1$	$61 \times 61 \times 96$
CCCP 1_2	$96 \times 1 \times 1$	$61 \times 61 \times 96$
Max - pooling	3×3, st. 2	$30 \times 30 \times 96$
Conv 2	$256 \times 7 \times 7$	$24 \times 24 \times 256$
CCCP 2_1	$256 \times 1 \times 1$	$24 \times 24 \times 256$
CCCP 2_2	$256 \times 1 \times 1$	$24 \times 24 \times 256$
max - pooling	3×3, st. 2	$11 \times 11 \times 256$
Modified Inception	Detailed in Fig. 1	$11 \times 11 \times 604$
Conv 3	$512 \times 3 \times 3$	$11 \times 11 \times 512$
CCCP 3_1	$512 \times 1 \times 1$	$11 \times 11 \times 512$
CCCP 3_2	$512 \times 1 \times 1$	$11 \times 11 \times 512$
Dropout	0.5	
Conv 4	$d \times 1 \times 1$	$11 \times 11 \times d$
Global Ave Pooling	11×11	$1 \times 1 \times d$

the training samples is enriched, which reduces overfitting. More importantly, the machine is guided in learning a greater number of invariant features, regardless of the existence of a certain local part.

3.2.1. Elastic meshing

Let us consider a gray-level image as a mapping, $I : \mathcal{D} \subset \mathbb{Z}^2 \rightarrow S$, where typically $S = \{0, 1, .., 255\}$. A naive method is used to divide the image into $L \times L$ equal rectangular regions; i.e., the area of each region is equal. We call this method a fixed meshing division. However, we determined experimentally that, in most cases, the randomly selected region is the background, which does not contain character information. This is because the centroid of a Chinese character image typically contains many strokes, whereas the boundary contains few strokes. To visualize this property, 3866 commonly used Chinese characters in 25 fonts were averaged. The final average image is shown in Fig. 2. Few strokes exist at the boundary, whereas many strokes exist at the centroid. It is interesting to note that the average character is not a Gaussian-like distribution, which was unexpected.

To address the uneven distribution of strokes or information in Chinese characters, an elastic meshing division method was developed. This elastic meshing technique has previously been applied to shape normalization in handwritten character recognition [37–39] and has been found to be a simple and effective preprocessing technique. Elastic meshing division for shape normalization makes it possible for each mesh to contain an equal distribution of the stroke histogram. In our system, elastic meshing is used to generate a set of regions adaptively, so that the regions contain equal amounts of information. The information is not limited to the geometric area, and it can be generalized to any type of feature. In this study, for simplicity, the information in each region was mea-

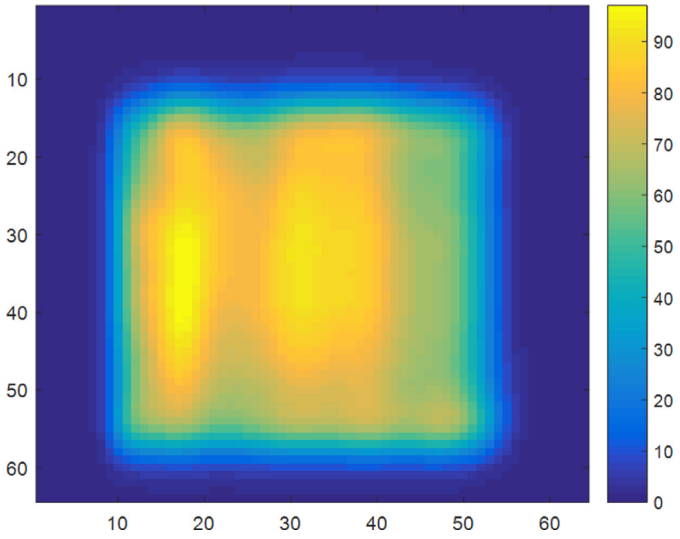


Fig. 2. Heat map of average of Chinese characters. Each character image was first inverted so that the background intensity was 0 and the stroke intensity was 255. The character images were then resized to 64×64. Finally, the element-wise average of these resized images was computed.

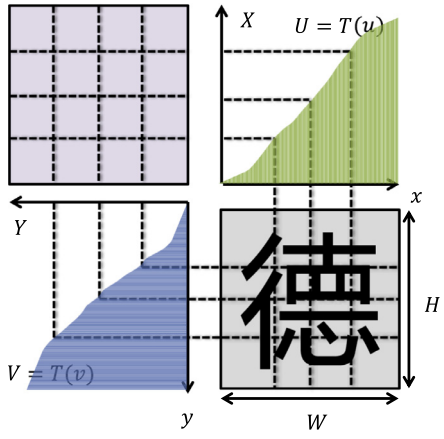


Fig. 3. Elastic meshing technique applied to one Chinese character.

sured by the accumulated pixel intensity. To equalize the accumulated pixel intensities in the regions, we first calculate the projection profile histogram on each axis. Fig. 3 illustrates the application of the elastic meshing technique to one character. Because the operation on the x axis is the same as that on the y axis, only the operation on the x axis is explained here. Let $\phi(x)$ denote the function for computing the projection profile histogram along the x axis, which is written as follows:

$$\phi(x) = \sum_{y=1}^{H} I(x, y), \tag{1}$$

where H is the image height. Next, the image is horizontally divided into L bars. We intend for the information in these bars to be equal. That is, each bar contains $\frac{1}{L}\sum_{x=1}^{W}\phi(x)$ information. Hence, the sum of the information from the first bar to the ith bar can be expressed as follows:

$$T(u_i) = \frac{i}{L}\sum_{x=1}^{W}\phi(x), \tag{2}$$

where $i \in \{1, .., L\}$ and u_i denotes the position along the x axis that corresponds to the ith bar. (Note that $\{u_1, u_2, \ldots, u_L\}$ is a set of breaking points that divides a character image into L bars.) $T(u_i)$ can be formulated as an accumulated distributed function as follows:

$$T(u_i) = \sum_{x=1}^{u_i}\sum_{y=1}^{H} I(x, y). \tag{3}$$

Finally, u_i is solved by substituting Eq. (2) with Eq. (1) and then substituting Eq. (3) with Eq. (2). In the same manner as $\{u_1, u_2, \ldots, u_L\}$ is determined, a set of breaking points along the y axis, namely, $\{v_1, v_2, \ldots, v_L\}$, can be determined.

Algorithm 1 Optimization for the DropRegion embedded IFN.

Require: Iteration number $t = 0$. Training dataset $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, $\boldsymbol{x}_n \in \mathbb{R}^D$ and $y_n \in \{0, 1, \ldots, K\}$. Learning rate is α_t.
Ensure:
Network parameters Θ.
repeat
$t \leftarrow t + 1$;
Randomly select a subset of samples from training set, namely a mini-batch.
for each training sample **do**
  if rand(0,1) $< \gamma$
    Perform elastic meshing division and obtain $L \times L$ regions;
    Randomly select a number of regions and disrupt these regions;
  end
  Perform forward propagation to obtain $\phi_n = f(\Theta, \boldsymbol{x}_n)$.
end for
$\Delta W = 0$.
for each training sample in a mini-batch **do**
Calculate partial derivative with respect to the output: $\frac{\partial J}{\partial \phi_n}$.
Run backward propagation to obtain the gradient with respect to the network parameters: $\Delta\Theta_n$.
Accumulate gradients: $\Delta\Theta := \Delta\Theta + \Delta\Theta_n$.
end for
$\Theta^t \Delta\Theta^{t-1} - \alpha_t \Delta\Theta$.
until converge.

3.2.2. DropRegion strategy

Inspired by Dropout [28], we designed a similar strategy that we called DropRegion. First, an image is selected. If a random number is greater than DropRatio, γ, the image requires further processing. Second, elastic meshing is performed on the image. Third, n ($n \geq 1$) local regions are randomly selected and disrupted by one type of noise. For simplicity, the randomly selected region is disrupted by multiplication using an all-zero mask. With these additional samples, the IFN can still be optimized in an end-to-end manner by stochastic gradient descent and back propagation. The proposed DropRegion algorithm is presented as Algorithm 1 at the end of this section.

To illustrate the proposed DropRegion algorithm, we consider a CNN with an input image $\mathbf{x}$, where $\boldsymbol{x} \in R^{d \times d}$. When DropRegion is applied, it can be written as $M \star \boldsymbol{x}$, where $\star$ is the element-wise product and M is a binary matrix that encodes the disruption information and can be expressed as follows:

$$\mathbf{M} = \begin{bmatrix} \mathbf{m}_{11} & \cdots & \mathbf{m}_{1L} \\ \vdots & \ddots & \vdots \\ \mathbf{m}_{L1} & \cdots & \mathbf{m}_{LL} \end{bmatrix}. \tag{4}$$

where $\boldsymbol{m}_{ij}$ is a sub-block matrix of $\boldsymbol{M}$. Note that some sub-block matrixes are randomly set as all-zero matrixes, whereas others are all-one matrixes. In other words, $\boldsymbol{M}$ has $C_{L\times L}^{n}$ types of patterns and $\boldsymbol{M} \sim U(0, 1)$. From that starting point, the DropRegion processes an input image $\mathbf{x}$ in a convolutional manner as follows:

$$\mathbf{x}_j^1 = g\left((\mathbf{M} \star \mathbf{x}) * \mathbf{k}_j^1 + \mathbf{b}^1\right), \tag{5}$$

where $\mathbf{x}_j^1$ is the jth feature map of the first convolutional layer, $\mathbf{k}_j^1$ is the convolutional kernel connected to the jth feature map, $\boldsymbol{b}^1$ is

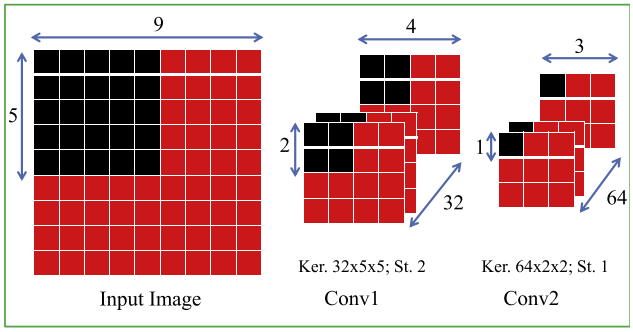


Fig. 4. Method by which DropRegion affects the convolutional layers. "Ker." represents the number of convolutional filters and their receptive field sizes ("num × size × size"), "St." is the stride size, and black pixels are zeroes.

the bias vector of the first convolutional layer, and $g(\cdot)$ is a nonlinear activation function. The receptive field corresponding to each unit of the feature map is a region in the input image. Let $\mathbf{r}$ denote this region. If $\mathbf{r}$ is disrupted by an all-zero mask, $\boldsymbol{m}$, and $\boldsymbol{b}^1$ is set as an all-zero vector, then $g((\boldsymbol{m} \star \boldsymbol{r}) * \boldsymbol{k}_j^1 + \boldsymbol{b}^1) = 0$. This is because many commonly used activation functions, such as the tanh, centered sigmoid, and rectified linear units (ReLU) [41] functions, have as a property $g(0) = 0$.

The above process is illustrated in Fig. 4. The 2×2 black region in each feature map of the *Conv1* layer is obtained by convolution with 32 kernels 5×5 in size. The region is then passed to an activation function. Similarly, the 1×1 black pixels in the feature maps of the *Conv2* layer are obtained by convolution with 64 kernels 2×2 in size. The pixels are then passed to an activation function. Note that the receptive field of the black units in the *Conv2* layer is the disrupted region in the input image. That is, DropRegion causes the neurons in the deeper layers to become zeroes. Accordingly, DropRegion can be regarded as a special case of Dropout; it differs, however, in that it acts on convolutional or pooling layers.

Given input data $\boldsymbol{x}$, the overall model, $f(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{M})$, outputs a result, $\boldsymbol{y}$, via a sequence of operations (e.g., convolution, nonlinear activation). The final value of $\boldsymbol{y}$ is obtained by summing over all possible masks, as follows:

$$y = \gamma f(\boldsymbol{x}; \boldsymbol{\theta}) + (1 - \gamma)\mathbb{E}_{\boldsymbol{M}}\left[f(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{M})\right], \tag{6}$$

where $\mathbb{E}_{\boldsymbol{M}}[f(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{M})]$ can be written as follows:

$$\mathbb{E}_{\boldsymbol{M}}\left[f(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{M})\right] = \sum_{\boldsymbol{M}} p(\boldsymbol{M}) f(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{M}). \tag{7}$$

Eq. (6) reveals the mixture model interpretation of DropRegion, where the output is a combination of $f(\mathbf{x}; \boldsymbol{\theta})$ and $f(\mathbf{x}; \boldsymbol{\theta}, \mathbf{M})$.

A connection exists between DropRegion and some previous data augmentation techniques. Krizhevsky et al. [12] enlarged a training set by performing tenfold cropping in one image and jittering the intensity of each image pixel. Bastien et al. [31] increased the richness of a training set using stochastic variations and noise processes (e.g., affine transformations, slant, etc.). Simard [32] expanded the training set for a neural network using an elastic transformation method. All of these methods have one characteristic in common, namely, they increase the training sample diversity and thus effectively reduce overfitting.

We can assess the augmentation capability of DropRegion as follows. Assume a character image is divided into $L \times L$ regions by an elastic meshing technique, and $n\,(n < L \times L)$ regions are randomly disrupted. The number of possible combinations of the remaining regions is then $C_{L\times L}^{n}$. We add all possible combinations to yield no more than n regions of disruption. We obtain a total of $\sum_{i=1}^{n} C_{L\times L}^{i} = 2^n$ variant samples. That is, the number of diverse samples is increased 2^nfold during the training phase, which demonstrates considerable augmentation capacity. As described previously, DropRegion disrupts the original topological structure of characters; nevertheless, it preserves the font information for later recognition. Intuitively, using these partially destroyed samples, the model is guided to capture more invariant representation, regardless of the presence or absence of one partial region.

4. Single-character-based font recognition

We observed that more than one font is applied in a text block in many Chinese documents. To highlight a particular document author's intention, certain characters in a sentence may appear in a different font. For example, specific characters may be printed in boldface, while others in the same sentence may appear in a regular typeface. To handle these situations and provide greater flexibility, it is necessary to design a font recognizer for single unknown Chinese characters.

The parameters of the IFN designed for single-character font recognition are listed in Table 1. The network is abbreviated as *SingleChar-IFN*. This network contains four convolutional layers, six CCCP layers, one modified inception module, and one global average pooling layer. The settings for each of the convolutional layers ("conv") are given in three sub-rows. The first represents the number of convolutional filters and their receptive field sizes ("num × size × size"), the second indicates the convolution stride ("st"), and the third denotes spatial padding ("pad"). When stride is equal to 1 or no padding is executed in the convolution operation, the corresponding "st" or "pad" sub-row is omitted. The same is true for the CCCP layer. The settings for the maximum pooling are specified as their kernel size ("size × size") and stride size ("st"). A similar arrangement is used for the global average pooling layer, except that no stride is necessary, as the pooling is globally executed over the entire feature map. The setting for the dropout is specified as the dropout rate, which is in fact a regularization operation executed over the adjacent lower layer. *SingleChar-IFN* takes 64×64 Gy images as input and outputs d dimensional vectors where d denotes the number of font categories. Unless otherwise stated, the parameter values for DropRegion in *SingleChar-IFN* were set to $L = 5$ and $\gamma = 0.5$.

5. Text block-based font recognition

The basic idea of text block-based font recognition is initial segmentation of the text block into separate characters and recognition of the single character font. To this end, a CFR system based on character segmentation is described. The original image is transformed into a binary image by an adaptive thresholding algorithm. Next, dilation and erosion operations are performed on the binary images to remove noise and connect the different parts of one character. In addition, horizontal and vertical projection profiles are successively applied. The goal of using the horizontal projection profile is to determine each text line, while that of using the vertical projection profile is to locate characters. Once the characters are located, font classification is performed on each character through *SingleChar-IFN*. The final result is obtained by averaging 30 confidence vectors. Multiple samples are used together to make the decision. This can be understood as an ensemble method that improves the model robustness. However, segmentation-based text block font recognition is limited in that it depends on the segmentation procedure and thus increases the complexity of recognition methods. To address this issue, we developed a segmentation-free font recognition system called *TextBlock-IFN*. A flowchart of the system is shown in Fig. 5. The parameters of *TextBlock-IFN* are outlined in Table 2. For the training phase, 128×128 patches are randomly cropped from one 384×320 text block image (a detailed

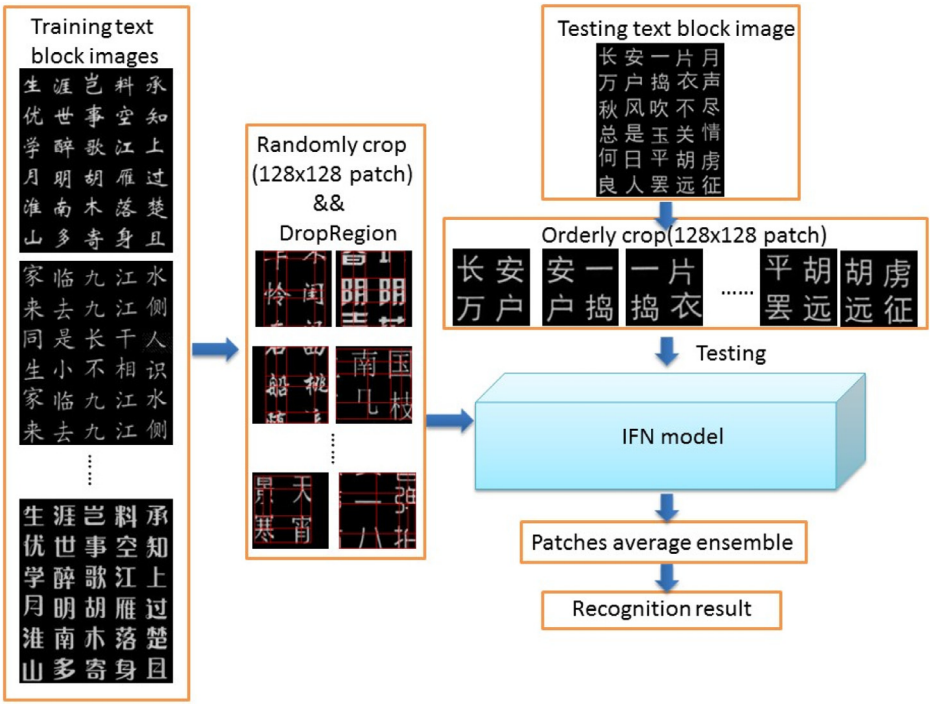


Fig. 5. Flowchart of segmentation-free text block font recognition.

description of the text block samples are given in Section 6.1). These 128 × 128 patches are then disrupted by the DropRegion method. In a manner similar to the process for training *SingleChar-IFN*, the DropRegion parameter values in *TextBlock-IFN* are set to $L = 5$ and $\gamma = 0.5$.

For the testing phase, we designed a multiple-decision fusion method combined with a sliding window approach. A sliding window was used to crop the image patches. The window size was 128 × 128, and the stride was 64 pixels. Using the sliding window, 20 cropped image patches were obtained from one text block. These cropped image patches were used as inputs to *TextBlock-IFN*. The output was a d-dimensional (d-D) confidence vector for each cropped patch. An averaging ensemble was adopted to enhance the model robustness. We applied element-wise accumulation to the d-D confidence vectors.

6. Experiments

6.1. Datasets

To the best of our knowledge, no publicly available datasets exist for Chinese font recognition. Thus, we propose two methods to construct both single character and text block Chinese font datasets. One is based on Microsoft Word software editing and image scanning to form neat synthetic datasets, and the other is a scene-guided text rendering method borrowed from [45] to form verisimilar real image datasets.

6.1.1. Neat synthetic datasets

We collected data samples from scanned documents to build a scanned Chinese font database, called SCF_DB_25. The database was constructed according to the following five steps. (1) A total of 3866 commonly used Chinese characters were edited with 25 font styles using the Microsoft Word software. The 25 fonts were Hei, YaHei, XiHei, YueHei, MeiHei, YaYuan, XingKai, Kai, FangSong, Song, ZhongSong, ZongYi, HuoYi, CuQian, GuangBiao, HuangCao, HuPo, LingXin, Shu, WeiBei, XinWei, YaoTi, YouYuan, LiShu, and ShuangXian. (2) The characters were then printed on paper and scanned as images. (3) The Hough line detector algorithm was used to align the images. (4) An image dilation method was used to connect adjacent characters. As a result, each text line was located using a horizontal projection profile, and each character was located using a vertical projection profile. (5) The character images, together with a number of annotations (including character category, font category, etc.), were saved in the database. After these steps were completed, we obtained a total of 96,650 font samples in SCF_DB_25. Furthermore, to validate the robustness of the proposed method for the dataset with much more variation, we extended the SCF_DB_25 to a much larger dataset called SCF_DB_280 that covered 280 Chinese font categories. The associated font information came from the collection of SCUT-SPCCI [22] created by our research group and the characters were the same as those in SCF_DB_25. As a result, there were a total of 1,082,480 samples in SCF_DB_280.

For text block-based font recognition, we also constructed a basic dataset called SCF_TDB_25 and an extended dataset called SCF_TDB_280. They included 25 and 280 Chinese font categories, respectively. A total of 320 blocks were composed for each font using 320 Tang poems, where one Tang poem corresponded to one text block consisting of six lines. Each line contained five characters. The characters not belonging to the set of 3866 characters were ignored. If the character number of one poem was less than 30, it was repeated from the beginning of the poem. Fig. 6 shows some synthetic text block samples from SCF_TDB _25. Also, we can calculate that total 8000 and 89,600 samples are included in SCF_TDB_25 and SCF_TDB_280, respectively.

6.1.2. Verisimilar real image datasets

Most attainable real-world text images do not have font label information, and the error-prone font labeling task requires font expertise that is out of reach of most people. Therefore, collecting and labeling real-world examples is very difficult [1]. However, recent work in [45] inspired us with a new solution for verisimilar real image font sample generation.

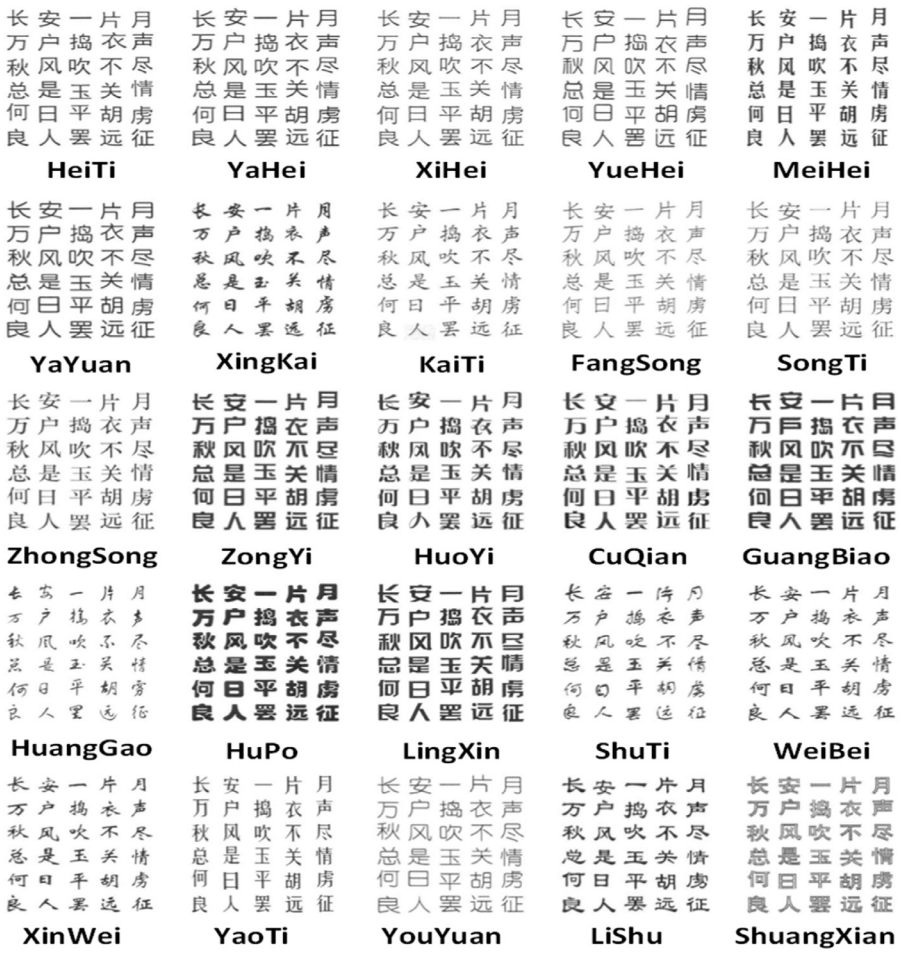


Fig. 6. Text block samples from SCF_TDB _25.

This font sample generation process started by acquiring the 3866 commonly used Chinese characters, as mentioned in Section 6.1.1, and downloading 8000 background images, as mentioned in [45]. These background images were extracted from Google Image Search using queries related to different objects/scenes, and indoor/outdoor and natural/artificial locales [45], exhibiting sufficient variation. Then the proposed image composition engine in [45] was applied to generate scene-text images, in which the Chinese character samples were blended into those background images using Poisson image editing [46]. During the process of blending, the character samples were rendered using our selected font, and transformed according to combined information from, for example, the local surface normal, dense pixel-wise depth map, and region's color of those background scenes. Next, the bounding boxes for each character were cropped from the synthetic scene text images to form single character font samples. Finally, visually almost invisible text samples were discarded using manual inspection. The filtered out samples composed a verisimilar real image dataset for Chinese characters called the Verisimilar Real Image Chinese Font Character dataset (VRICFChar). Fig. 7 shows examples from VRICFChar. From this figure, it can be observed that the generated font images have large appearance variations, including scaling, background clutter, lighting, noise, perspective distortions, orientations, and compression artifacts [1]. All the variations were inherited from the various scene images in the procedure of text rendering because this procedure was guided by the background images. Complex interactions of text content with background were also integrated in the generating process to make the font samples nearly real images. A total of 30 font categories were used, of which 25 came from the dataset mentioned in Section 6.1.1. Among the 25 categories, only "HuangCao" and "Shu" were handwritten typefaces; the remaining 23 categories were all printed. To verify the robustness of the proposed method for handwritten font types, we added five well-known handwritten typefaces to form the verisimilar real image font datasets, that is, Chenjishi, Chenxudong, Jingboran, Maozedong, and Xujinglei. Similarly, we constructed a text block font dataset, VRICFTextblock. Source text for VRICFTextblock came from the aforementioned 320 Tang poems instead of single Chinese characters. Fig. 8 shows examples of real text block font images.

Additionally, Table 3 provides the statistics of VRICFChar and VRICFTextblock. It can be observed that the font sample number varied with different font types. Moreover, it can be calculated that 286,987 and 21,546 samples were included in the VRICFChar and VRICFTextblock datasets, respectively, by setting the maximum number of allowable segmented regions to five in the blend procedure.

6.2. Data preprocessing and experiment setting

6.2.1. Data preprocessing

All the single character font samples from neat synthetic datasets were resized to 60×60. Then, each image was padded with two pixels in the boundary; hence, the input image for SingleChar-IFN was 64×64. All the text block font samples were resized to 384×320, which was well matched in proportion to

Fig. 7. Font samples of a single Chinese character from the VRICFChar dataset.

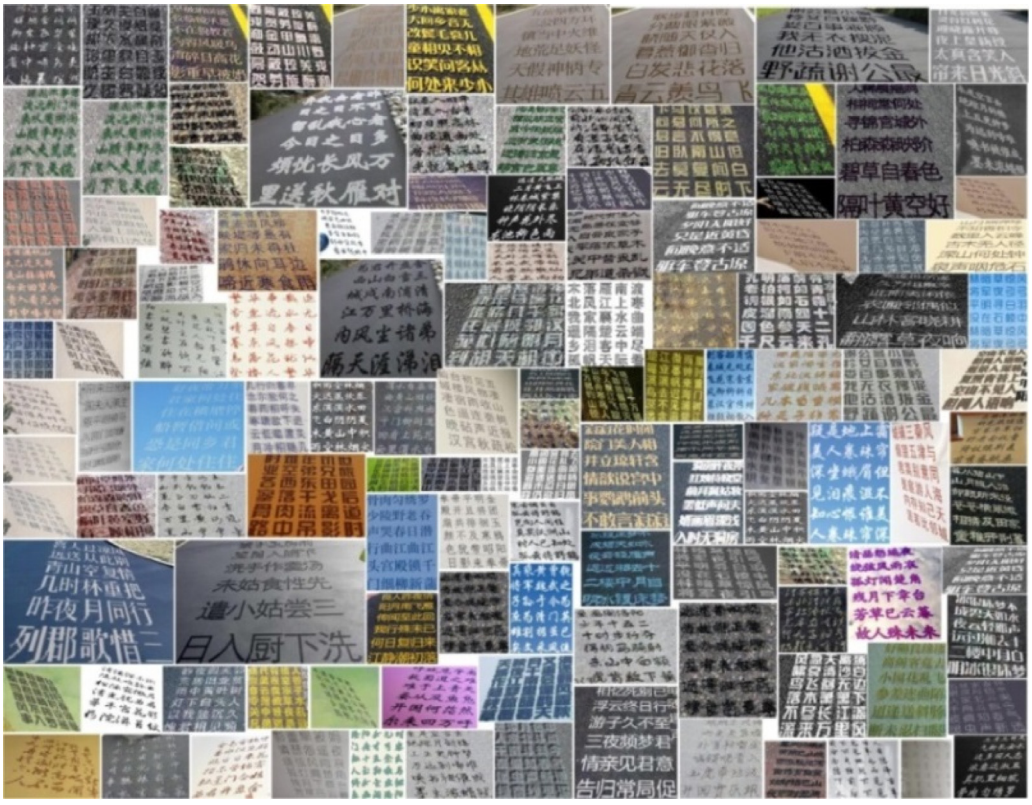


Fig. 8. Sample images of text blocks from the VRICFTextblock dataset.

Table 3
Font sample numbers for VRICFChar and VRICFTextblock.

Font Type	SRICFChar	SRICFTextblock	Font Type	SRICFChar	SRICFTextblock
Hei	11,025	672	HuPo	9873	626
YaHei	8423	300	LingXin	7264	782
XiHei	8781	251	WeiBei	10,844	862
YueHei	10,511	589	XinWei	9611	1139
MeiHei	10,355	762	YaoTi	10,191	968
YaYuan	10,685	524	YouYuan	9056	562
XingKai	8772	982	LiShu	8178	783
Kai	8956	349	ShuangXian	6701	697
FangSong	11,682	792	HuangCao	7875	516
Song	8543	310	Shu	13,595	1404
ZhongSong	10,105	303	Chenjishi	11,665	639
ZongYi	11,483	570	Chenxudong	6656	1214
HuoYi	10,228	457	Jingboran	6854	661
CuQian	8834	855	Maozedong	10,235	1147
GuangBiao	10,776	1046	Xujinglei	9230	784

a single character because each Tang poem had six lines and five columns.

Different from the neat synthetic data, more sophisticated preprocessing steps had to be performed on the verisimilar real image datasets because the images from these datasets typically had much larger appearance variations, for example, scaling, background clutter, lighting, noise, perspective distortions, orientations, and blur. The preprocessing process is described as follows: First, both single character and text block font images were tilt corrected using affine transformation. Then, they were separately enhanced on their RGB color channels by means of histogram equalization. The three enhanced color channels were recombined to form a new color image. Then, single character sample images were normalized into 64×64 and text block font images were normalized into 384×320. Finally, all the color samples were converted into grayscale as the input to *SingleChar-IFN* or *TextBlock-IFN*.

6.2.2. Experiment setting

All the training/testing splits were performed according to the source text for each font category; that is, there were no common Chinese characters or Tang poems in the training and testing datasets for convincing all the experiments. Specifically, each source character or Tang poem corresponded to one sample image for each font in the neat synthetic datasets. Therefore, the sample size for each font was equal to the number of source texts, for example, 3866 samples per font in SCF_DB_25 and SCF_DB_280; and 320 samples per font in SCF_TDB_25 and SCF_TDB_280. The situation was different for the verisimilar real image datasets. Each source text, regardless of whether it was a single character or text block, generally corresponded to several font samples. Therefore, a random training/testing split according to the source text could result in a difference in sample size used for training and testing. We denote the source text number for training by *TrNum* and testing by *TsNum*.

We implemented our algorithm based on the Caffe [42] framework. The learning rate was updated as follows:

$$lr = base\_lr \times \left(1 - \frac{iter}{max\_iter}\right)^{factor}, \tag{8}$$

where ***base_lr***, **max_*iter***, and **factor** were set to 0.01, 1.5e5, and 0.5, respectively. The momentum was set to 0.9, and the weight decay rate was set to 2e-4.

6.3. Results

6.3.1. Single-character-based font recognition

This part of the experiment was designed to achieve two goals. One was to analyze how the behavioral characteristics of the proposed DropRegion-IFN were affected by the IFN network structure, DropRegion training strategy, and elastic mesh technique. On the aforementioned basis, a comparison with three baseline methods was made to validate the overall performance. The corresponding experiments were conducted on SCF_DB_25. The other goal was to investigate whether the proposed method was sensitive to font class size and still effective for verisimilar real font images. Experiments for this purpose were conducted on SCF_DB_280 and VRICFChar, respectively.

6.3.1.1. Evaluation of the effect of IFN network structure. We evaluated the effectiveness of the proposed IFN network structure in contributing to the overall performance achieved in Chinese character font recognition. Four baseline network architectures, summarized as shown in Fig. 9, were used for comparison. The four are denoted Basic-CNN, DeepFontNoD, MDLSTM [43] and GoogleNet [18]. Basic-CNN is a standard CNN architecture consisting of four stacked convolutional layers, optionally followed by max-pooling, and then, ending with a full connection layer feeding into a final softmax layer. The architectural parameters of Basic-CNN are the same as that of IFN at the corresponding layer. DeepFontNoD has the same basic network structure as the DeepFont system [1], with low-level unsupervised and high-level supervised subnetwork decomposition removed. This structure is similar to that of AlexNet [12], which serves as an example for the adoption of a conventional CNN structure instance in the field of font recognition. For MDLSTM, we adopted the same network hierarchy by repeatedly composing MDLSTM layers with feed-forward layers, as in [43]. GoogleNet has almost the same settings as in [18], except that the input size is different (64×64 vs. 224×224). The hyperparameters for the last average pooling layer were adjusted from "7×7, st. 1" to "2×2, st. 1," as shown in Fig. 9. The dashed rectangle in the figure represents a repetitive network building block, which can be removed from the corresponding network to obtain a simplifier model for adjusting to training sample scale. For example, when $TrNum = 200$ samples were chosen from each font to build a training set, only one NIN building block was retained in the subsequent experiment setting to adapt to the case of a small sample size. Two NINs were retained in the case of $TrNum = 400$, and three were retained in the case of $TrNum = 1000$. A similar adjustment to the building block size was adopted for Basic-CNN when comparison experiments were conducted for the purpose of verification of the network structure design.

Table 4
Recognition accuracies (%) of IFN and other baseline networks (Basic-CNN, DeepFontNoD, MDLSTM, and GoogleNet) for single-character images.

TrNum	IFN	Basic-CNN	DeepFontNoD	MDLSTM	GoogleNet
200	**89.34**	83.64	73.00	73.52	80.57
400	**91.54**	88.06	80.51	78.20	86.05
1000	93.59	92.85	90.00	81.28	**94.98**

To ensure a fair comparison, we used Dropout for all the models, rather than DropRegion. The experimental results for the task of single-character-based font recognition are summarized in Table 4. The results show that the IFN architecture almost always outperformed all four of the baseline networks for each font for sample sizes of 200, 400, and 1000. The results confirm the effectiveness of our IFN design. As Table 4 shows, Basic-CNN performs much better than DeepFontNoD, even they are homogenous. This may be because Basic-CNN naïvely adjusts the model size by adding or removing building blocks to or from the network structure to adapt to the sample scale. This model size adjustment scheme was adopted in both IFN and Basic-CNN, but IFN results in a clear improvement in the accuracy in comparison to Basic-CNN. Therefore, it is of critical importance that the architectural choice is elegantly designed for the specific font recognition task. Compared to MDLSTM, the relative increase in font category recognition accuracy achieved by IFN is up to 22% when $TrNum = 200$. The accuracy of GoogleNet was greater than our IFN by approximately 1.39% when $TrNum = 1000$. However, the IFN network has many fewer parameters than GoogleNet (20 M vs. 34 M). To summarize the comparative evaluation of the five various architectures, the design of our IFN architectural elements is advantageous to Chinese font recognition.

6.3.1.2. Evaluation of DropRegion in comparison to Dropout. We evaluated the regularization effect of Dropout and DropRegion using *SingleChar-IFN*. In the experiments, each single image was divided into 5×5 regions, and a maximum of 13 regions were randomly dropped. The results are shown in Table 5. It can be seen that Dropout improved the accuracy by 4.18%, 1.36%, and 0.01% when 200, 400, and 1000 training samples, respectively, were used

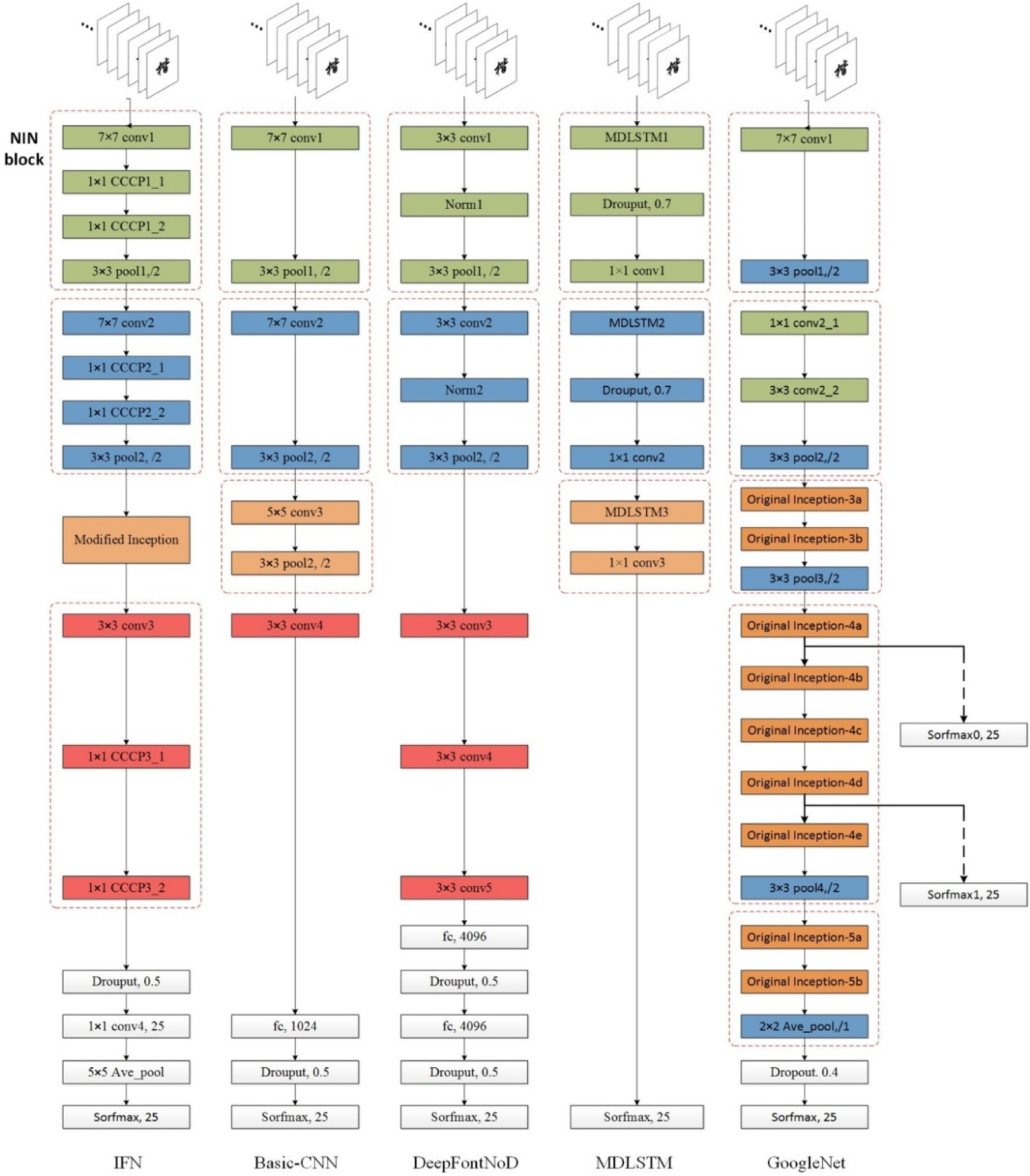


Fig. 9. Five network architectures used in our experiments. This plot shows the network structure for the five different baseline models.

for each font. DropRegion boosted the accuracy by 12.10%, 7.74%, and 5.27%, respectively, for the same numbers of training samples, exhibiting better regularization performance than Dropout. We also investigated the performance of a combination of DropRegion and Dropout. As Table 5 shows, the combination further improved the classification accuracy. This suggests that Dropout and DropRegion are complementary when used together. It is worth mentioning that the recognition accuracies we achieved exceeded that achieved with Tao's method [43], even when we used training sets only half the size of those used with Tao's method (e.g., 2000 for each font), based on total data sets of almost the same size.

6.3.1.3. Evaluation of fixed and elastic mesh techniques. We evaluated three different methods using training samples of different sizes. The methods were *SingleChar-IFN, SingleChar-IFN*+DropRegion (fixed mesh), and *SingleChar-IFN*+DropRegion

Table 5
Font recognition accuracy (%) of DropRegion and Dropout on single characters.

TrNum	None	Dropout	DropRegion	DropRegion+Dropout
200	85.16	89.34	97.26	**97.31**
400	90.18	91.54	97.92	**98.63**
1000	93.59	93.60	98.86	**98.98**

Table 6
Recognition accuracies (%) of different methods for single character images. *TrNum* is the number of training samples for each font.

TrNum	*SingleChar-IFN*	*SingleChar-IFN*+ DropRegion (fixed mesh)	*SingleChar-IFN*+ DropRegion (elastic mesh)
200	89.34	97.20	**97.31**
400	91.54	98.30	**98.63**
1000	93.59	98.96	**98.98**

(elastic mesh). The value of the parameter L was fixed or the elastic meshing division was set to 5, and the value of the parameter γ was set to 0.5.

The experimental results are presented in Table 6. For *TrNum* = 200, 400, and 1000, the DropRegion method with a fixed mesh improved the recognition accuracy by 7.86%, 6.76%, and 5.37%, respectively. The DropRegion method with an elastic mesh improved the recognition accuracy by 7.97%, 7.09%, and 5.39%, respectively. These results suggest that, regardless of whether the fixed or elastic meshing method is used, DropRegion improves the capacity of representation learning, especially when the training samples are small. These two meshing techniques are both effective, but the results suggest that the elastic meshing method is slightly superior to the fixed meshing method.

6.3.1.4. Investigation of number of dropped regions. We investigated the effect of the number of dropped regions. *TrNum* is set as 200. Each single character image is divided into 5×5 regions. The maximum number of dropped regions, denoted as n, is varied from 1 to 24. Different n represent different increased folds of the variant sample. The experimental results are shown in Table 7. The best recognition accuracy (97.31%) is achieved when the number of dropped regions is 13. When the number of dropped regions is less or more than 13 (between 1 and 24), the recognition accuracy is much lower than 97.31%. This is because few stochastic variations are introduced when the number of dropped regions is decreased, and the font information is not sufficient for learning a robust model when the number of dropped regions increases.

6.3.1.5. Investigation of the number of divided regions. We also investigated the effect of the number of divided regions. *TrNum* was set to 200. Each single character image was divided into $L \times L$ regions, where $L = [3, 5, 7, 9, 11, 13]$. The maximum number of dropped regions, denoted as n, was set to $\frac{L \times L}{2}$. The experimental results are shown in Table 8. It can be observed that the highest accuracy of 97.31% was achieved when $L = 5$. However, recognition accuracy fluctuated only slightly with the varied number of divided regions. The results are reasonable because L only regulated the precision level of the DropRegion operation. This was true in particular when n was fixed at $\frac{L \times L}{2}$, that is, always approximately half of the divided regions were dropped during training in the maximum case.

6.3.1.6. Comparison with the baseline methods. We compared performance between the proposed DropRegion-IFN and three baseline methods, that is, LBP+SVM, BIF+DGPP [47], and C1+LPP [48]. We set $\boldsymbol{L} = 5$ and $\boldsymbol{n} = 13$ for DropRegion-IFN for optimal performance. For LBP+SVM, we first computed the uniform pattern for each pixel in the single character image from a 3×3 pixel neighborhood. Then, we counted the histogram of all types of uniform patterns and normalized the histogram. Thus, an LBP [49] descriptor with 59 dimensions was obtained. Finally, a linear SVM [50] classifier was trained to predict the Chinese character font. In BIF+DGPP [47], the high-dimensional biologically inspired feature (BIF) was mapped to a low-dimensional space using manifold learning algorithm DGPP [47] (i.e., the dimensionality was equal to 150), and the pairwise multiclass SVM with a Gaussian kernel was applied for final font classification. The C1 unit of the BIF was extracted using the same parameters as in [47]. However, we extracted intensity and color feature maps using $\mathbf{c} = 1, 2, 3$ for center levels and $\mathbf{s} = \mathbf{c} + \mathbf{d}$, with $\mathbf{d} = 2$ for surrounding levels instead of the parameters used in [47]. This was because of the small size of the single character image, for example, 64×64. Therefore, the BIF contained 25 feature maps, that is, three intensity feature maps, six color feature maps, and 16 C1 units feature maps. In C1+LPP [48], we used the same feature maps as those in BIF+DGPP and reduced feature dimensionality to 24 using LPP [48]. The nearest neighbor rule was used for the final font classification. Additionally, the optimal SVM regularization parameters for LBP+SVM and BIF+DGPP [47] were set to 3.5 and 7, respectively.

Table 9 summarizes the comparison results for the cases when *TrNum* = 200, 400, and 1000. From the table, our proposed DropRegion-IFN method significantly outperformed all three baselines in terms of classification accuracy in all cases. Specifically, the DropRegion-IFN improved by approximately 7% compared with BIF+DGPP in the minimum case when *TrNum* = 1000. In the maximum case, DropRegion-IFN exceeded LBP+SVM by nearly 28% when *TrNum* = 200. Among the three baseline methods, two manifold learning-based dimensionality reduction algorithms for font categorization obtained essentially the same level of performance, and LBP+SVM performed the worst.

Furthermore, the runtime performance was analyzed. For a fair comparison, all the experiments were conducted on a 4.0 GHz Intel i7 CPU with Ubuntu 16.04 and NVIDIA GeForce GTX 1080 GPU. In Table 10, we provide the average test time per sample for both the proposed DropRegion-IFN and all the aforementioned baseline methods. For this table, the test time of the baseline algorithms was calculated based on the CPU running. Specifically, the running time of two manifold learning based methods involved BIF feature extraction, dimensionality reduction using DGPP or LPP, and prediction of the SVM or NN classifier. The test time for LBP+SVM was equal to the total time cost of LBP feature extraction and SVM prediction. Differently, we simply ran our deep IFN on new font images to obtain the score, and further to make font prediction, by which the process time consumed was the running time of DropRegion-IFN. Time calculations were performed for the two cases of running on CPU and GPU respectively. Running DropRegion-IFN on a CPU was for a fair comparison with the baseline algorithms. From Table 10, running DropRegion-IFN on a CPU cost slightly more time than each of the three baseline algorithms. However, the running time of DropRegion-IFN on a GPU dropped dramatically by two orders of magnitude in comparison with the three baseline methods. Among the three baseline methods, LBP+SVM cost the least time because of its fast LBP feature extraction.

6.3.1.7. Investigation of robust performance against the dataset scale

We justified the adaptability of our method to a much larger font class size by conducting experiments on SCF_DB_280. A similar experimental protocol to that in Section 6.3.1.2 was adopted for the investigation, that is, *TrNum* = 200, 400, or 1000 for each font were selected for training and the remainder for testing. Experimental results using IFN with and without the DropRegion strategy are listed in Table 11 for comparison. Unexpect-

Table 7
Evaluation of recognition accuracy (%) achieved by varying the number of dropped regions.

n	1	2	3	4	5	6	7	8
Accuracy	90.16	94. 46	95.51	95.80	95.98	96.19	96.28	96.73
n	9	10	11	12	13	14	15	16
Accuracy	96.89	96.97	97.12	97.20	**97.31**	96.92	96.80	96.67
n	17	18	19	20	21	22	23	24
Accuracy	96.59	96.35	96.09	95.94	95.87	94.82	93.84	93.49

Table 8
Evaluation of recognition accuracy (%) achieved by varying the number of divided regions.

L	3	5	7	9	11	13
Accuracy	96.84	**97.31**	96.97	96.97	96.31	95.58

Table 9
Recognition accuracy (%) of DropRegion-IFN and three baseline methods.

TrNum	DropRegion-IFN	LBP+SVM	BIF+DGPP[47]	C1+LPP[48]
200	**97.31**	69.57	89.56	90.79
400	**98.63**	72.57	91.19	91.40
1000	**98.98**	75.73	92.47	92.10

Table 10
Runtime comparison between DropRegion-IFN and the three baseline methods.

Method	Average testing time per sample (ms)
DropRegion-IFN (GPU)	0.789
DropRegion-IFN (CPU)	126.0
LBP+SVM	16.28
BIF+DGPP[47]	88.29
C1+LPP[48]	85.91

Table 11
Evaluation of recognition accuracy (%) achieved on SCF_DB_280.

TrNum	IFN	DropRegion-IFN
200	97.44	98.98
400	98.51	99.11
1000	98.75	99.17

edly, the proposed IFN and DropRegion-IFN worked well in all cases. When *TrNum* = 1000, IFN achieved high accuracy of up to 99.17% with DropRegion and 98.75% without DropRegion. Even if the total training samples accounted for only slightly more than 5% of the total dataset, corresponding to the case of *TrNum* = 200, DropRegion-IFN achieved high recognition accuracy of 98.98%.

6.3.1.8. Performance with verisimilar real font images. We further evaluated the effectiveness of the proposed DropRegion-IFN method with verisimilar real images on the VRICFChar dataset. One thousand random Chinese characters were selected for training and the remainder used for testing. The training and testing sample size for each font determined by a random character split is shown in Fig. 10. From the figure, a total of 74, 441 samples were used for training and 212, 546 for testing.

Fig. 11 shows the classification confusion matrix of the 30 verisimilar real image font categories. It can be observed that 35% of WeiBei test samples were classified into XinWei, whereas 24% of XinWei test samples were misclassified as WeiBei. This could be explained by their visual similarity, shown in Fig. 12(a). Additionally, the Chenxudong typeface achieved the worst accuracy of 51% among all 30 categories. Fig. 12(b) visualizes some images that correspond to Chenxudong and its three most easily confused fonts, Chenjishi, Jingboran, and Xujinglei, which demonstrates the subtle inter-class variations among them. The misclassification of Chenxudong as Chenjishi, HuangCao, and Xujinglei was as high as 8%, 8%, and 15%, respectively. Furthermore, HuPo achieved the highest accuracy of 97%. The total average accuracy for each font category reached as high as 90.02%, which shows that the proposed the method is very promising for Chinese font recognition with verisimilar real images.

Furthermore, Fig. 12(c) and (d) show example test images that were correctly and incorrectly classified by our method, respectively. Remarkably, our model was robust to a cluttered background, slight amount of distortion or orientation, and lighting variation, shown in Fig. 12(c). However, in the case of serious text degradation, for example, very low resolution, extremely noisy input, and much cluttered background, shown in Fig. 12(d), the algorithm failed.

6.3.2. Segmentation-free-based text block font recognition

6.3.2.1. Results on SCF_TDB_25. We performed eight sets of experiments to evaluate segmentation-free text block font recognition on SCF_TDB_25. We followed the dataset partition scheme introduced by Tao et al. [43] The results were shown in Table 12. An average accuracy of 99.78% was achieved when *TrNum* and *TsNum* were 30 and 10, respectively, and an average accuracy of 99.28% was achieved when *TrNum* and *TsNum* were 20 and 20, respectively.

We compared our segmentation-free recognition system with the segmentation-based system described in Section 5 and four representative state-of-the-art systems, LBP+SVM, Gabor+WED [3], sparse discriminative information preservation [10] (SDIP), and marginal Fisher's analysis [33] (MFA). As Table 13 showed, the proposed segmentation-free-based method achieved the highest accuracy of 99.78% when the training number (*TrNum*) was 30 and the test number (*TsNum*) was 10. Similarly, the highest accuracy of 99.28% was achieved when *TrNum* and *TsNum* were both 20. These results can be attributed to the design of the deep IFN and the design of DropRegion. The proposed segmentation-based method achieved 96.90% and 96.80% accuracy for two different training samples, thereby ranking second among the systems. Compared with other handcrafted features (i.e., LBP and Gabor features), deep IFN automatically discovers the features of font recognition within an end-to-end learning framework, which proves to be a powerful method for feature learning. The proposed DropRegion method improves the performance of IFN because it significantly increases the number of training samples and can be regarded as an effective regularization technique.

6.3.2.2. Results on SCF_TDB_280. We justified the adaptability to font category scale by conducting experiments on the SCF_TDB_280 dataset. Table 14 shows the results for the cases of *TrNum/TestNum* = 160/160 and *TrNum/TestNum* = 240/80. As shown by the high accuracies of approximately 97% for both cases, the proposed segmentation-free text block font recognition method adapted well to the large font classes.

6.3.2.3. Results on VRICFTextblock. We evaluated the performance of the proposed segmentation-free-based text block font

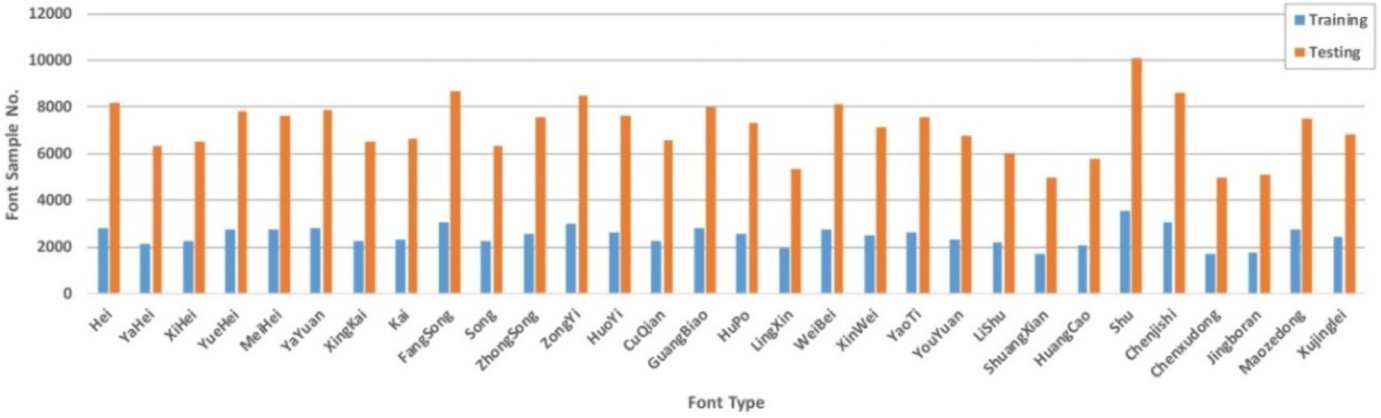


Fig. 10. Training and testing sample size for each font denoted as "(Font Class Index) Font name (Training No., Testing No.)". (1) Hei (2836, 8189); (2) XiHei (2124, 6299); (3) YueHei (2261, 6520); (4) MeiHei (5736, 7775); (5) MeiHei (2726, 7629); (6) YaYuan (2802, 7883); (7) XingKai (2262, 6510); (8) Kai (2340, 6616); (9) FangSong (3025, 8657); (10) Song (2250, 6293); (11) ZhongSong (2552, 7553); (12) ZongYi (2986, 8497); (13) HuoYi (2606, 7622); (14) CuQian (2274, 6560); (15) GuangBiao (2794, 7982); (16) HuPo (2576, 7297); (17) LingXin (1913, 5351); (18) WeiBei (2727, 8117); (19) XinWei (2516, 7095); (20) YaoTi (2630, 7561); (21) YouYuan (2317, 6739); (22) LiShu (2170, 6008); (23) ShuangXian (1715, 4986); (24) HuangCao (2082, 5793); (25) Shu (3526, 10,069); (26) Chenjishi (3065, 8600); (27) Chenxudong (1696, 4960); (28) Jingboran (1773, 5081); (29) Maozedong (2737, 7498); (30) Xujinglei(2424, 6806).

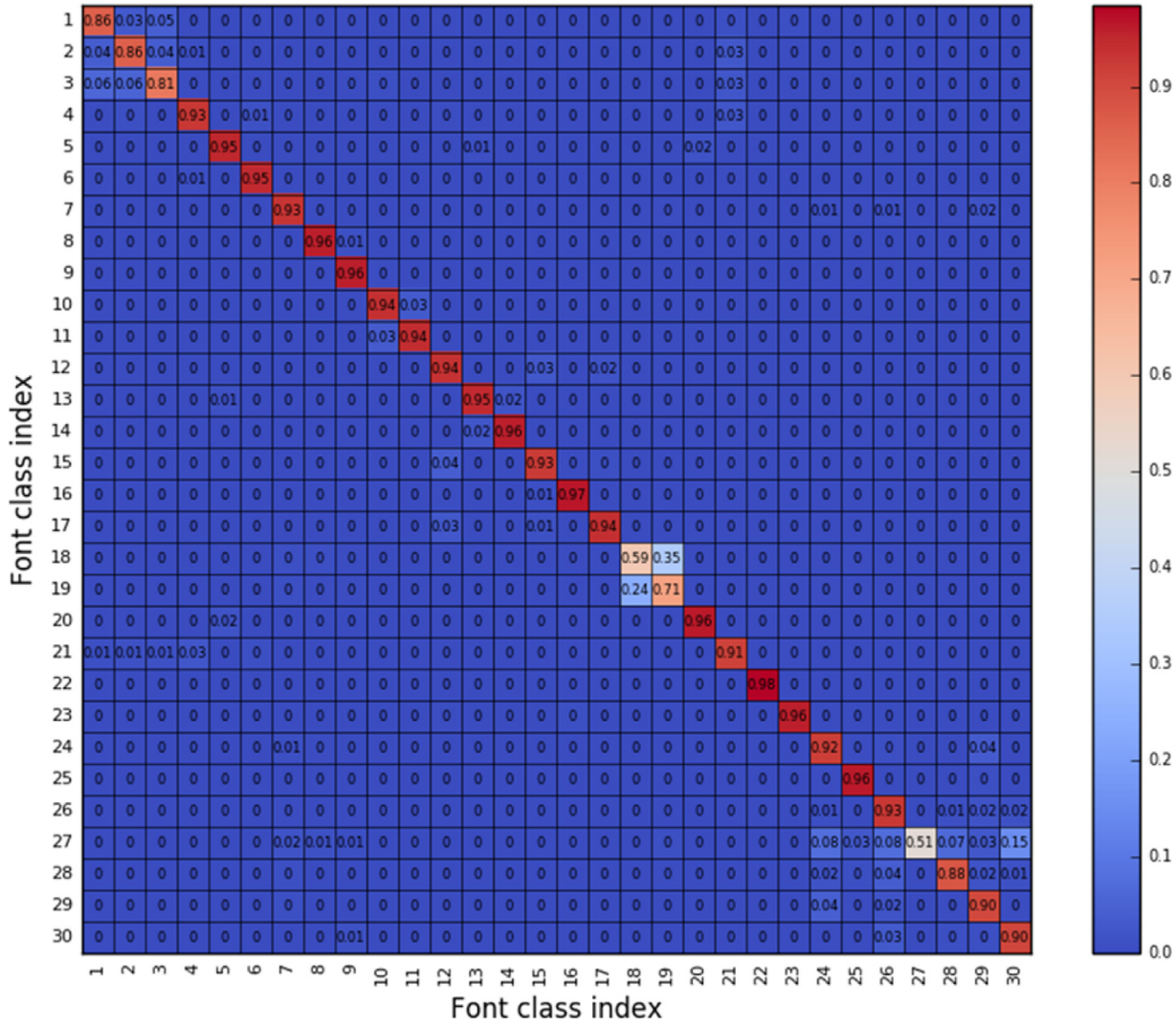


Fig. 11. Confusion matrix for 30 font classes of VRICFChar indexing from 1 to 30.

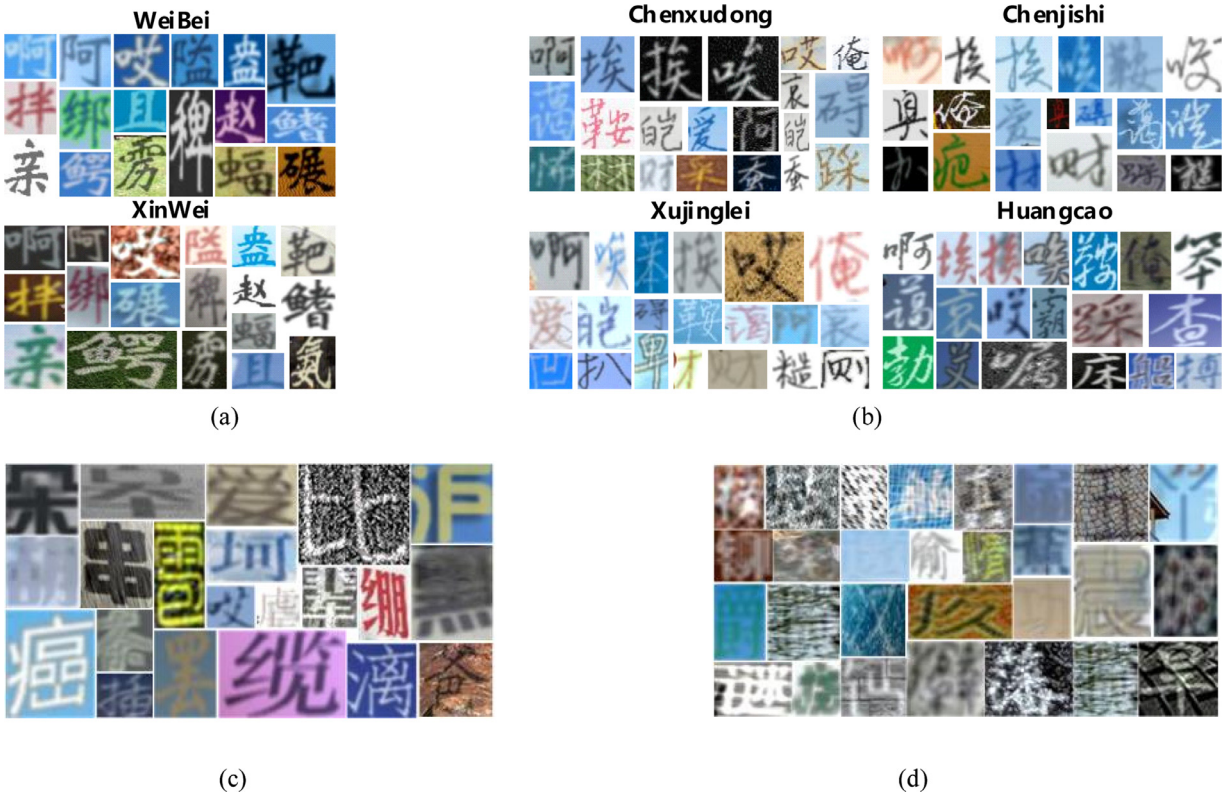


Fig. 12. (a) Samples from easily confused typefaces WeiBei and XinWei. (b) Samples from Chenxudong and the other three easily confused typefaces: Chenjishi, Xujinglei, and HuangCao. (c) Test samples that were correctly classified. (d) Test samples that were incorrectly classified.

Table 12
Recognition accuracy (%) of multiple groups of segmentation-free text block font recognition experiments. G1–G8 denote eight groups of experiments. *TrNum* is the number of training samples for one font. *TsNum* is the number of test samples for one font.

Dataset	G1	G2	G3	G4	G5	G6	G7	G8	Average
TrNum = 30 *TsNum* = 10	99.6	99.92	99.9	99.9	99.96	99.58	99.66	99.72	99.78
TrNum = 20 *TsNum* = 20	99.57	99.4	99.29	98.93	99.82	99.6	98.3	99.3	99.28

Table 13
Recognition accuracy (%) of the proposed method and other state-of-the-art methods.

Methods	*TrNum* = 30 *TsNum* = 10	*TrNum* = 20 *TsNum* = 20
Proposed (segmentation-free)	**99.78**	**99.28**
Proposed (segmentation)	96.90	96.80
LBP+SVM	95.35	94.20
Gabor+WED [3]	95.45	94.48
SDIP [10]	93.00	91.60
MFA [33]	91.60	90.10

Table 14
Evaluation of recognition accuracy (%) achieved on SCF_TDB_280.

TrNum/TsNum	DropRegion-IFN
160/160	96.92
240/80	97.33

recognition on verisimilar real image dataset VRICFTextblock. Two training/testing split schemes were adopted, that is, *TrNum/TsNum* = 160/160 and *TrNum/TsNum* = 240/80. Each split scheme was replicated five times randomly. Table 15 reports the averages and standard deviations of the training font sample size and recognition accuracy. It can be observed that the deviation of recognition accuracy was very small, even when the number of font samples fluctuated for a different random split. The results show the semantic invariance of the proposed font recognition method.

Table 15
Training font sample size and recognition accuracy (%) on VRICFTextblock.

TrNum/TsNum	Training font sample #	Accuracy
160/160	10,743 ± 92	97.99 ± 0.034
240/80	16,039 ± 288	98.42 ± 0.109

7. Conclusions

In this paper, we presented a new method called DropRegion-IFN, for Chinese font recognition. This method highlights the elegant design of a deep font network, namely IFN, which integrates a modified inception module, CCCP layers, and global average pooling. DropRegion, a new data augmentation and regularization technique, is proposed for seamless embedment in the IFN framework, enabling an end-to-end training approach and enhanc-

ing model generalization. Furthermore, we proposed two methods to construct both single character and text block Chinese font datasets. One was based on Microsoft Word software editing and image scanning to form four neat synthetic datasets, including 25-category basic datasets and 280-category extended datasets; and the other was a scene-guided text rendering method to form verisimilar real image datasets VRICFChar and VRICFTextblock. Extensive experiments were conducted on all six datasets. The recognition accuracies on the neat synthetic datasets were all over 97%, using either single-character-based or segmentation-free text block-based font recognition. With regard to verisimilar real image datasets, an accuracy of 90.02% on VRICFChar and approximately 98% on VRICFTextblock were achieved, which demonstrates that the proposed method was very promising for Chinese font recognition with real images.

In future research, we will construct a larger verisimilar real image Chinese font dataset, e.g., 280 categories, considering more practical scenarios of applications. For example, each text block would contain a different number of Chinese characters; the size of each character in the text block would vary greatly, and the character spacing in the same text block would be different. A sharp increase in class numbers together with the complex diversity just mentioned will result in greater challenges for font recognition with verisimilar real images. Additionally, we particularly intend to collect a small number of labeled real-world text images and determine whether there is any difference between the verisimilar font samples and manually collected real images. Finally, we would like to extend the DropRegion-IFN method to handle the practical cases in which font class samples are scarce.

Acknowledgments

This work was supported by NSFC [grant number: 61472144, 61673182], the National Key Research & Development Plan of China [grant number: 2016YFB1001405], GDSTP [grant number: 2014A010103012, 2015B010101004, 2015B010130003, 2015B010131004, 2014A020208112, 2016A010101014] and GDUPS [2011], Science and Technology Program of Guangzhou, China [grant number: 201707010160].

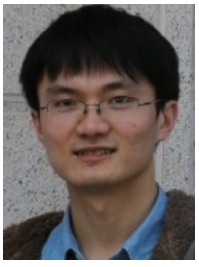
References

[1] Z. Wang, J. Yang, H. Jin, E. Shechtman, A. Agarwala, DeepFont: identify your font from an image, in: Proceedings of ACM MM, 2015, pp. 451–459.
[2] M. Lutf, X. You, Y. Ming Cheung, C.P. Chen, Arabic font recognition based on diacritics features, Pattern Recognit. 47 (2) (2013) 672–684.
[3] F. Slimane, S. Kanoun, J. Hennebert, A.M. Alimi, R. Ingold, A study on font–family and font-size recognition applied to Arabic word images at ultra-low resolution, Pattern Recognit. Lett. 34 (2) (2013) 209–218.
[4] B. Bataineh, S.N.H.S. Abdullah, K. Omar, A novel statistical feature extraction method for textual images: optical font recognition, Expert Syst. Appl. 39 (5) (2012) 5470–5477.
[5] Z. Yang, L. Yang, C.Y. Suen, A new method of recognizing Chinese fonts, in: Proceedings of ICDAR, 2015, pp. 962–966.
[6] R. Cooperman, Producing good font attribute determination using error-prone information, in: Electronic Imaging'97, Proceedings of SPIE, 1997, pp. 50–57.
[7] A. Zramdini, R. Ingold, Optical font recognition using typographical features, IEEE Trans. PAMI 20 (8) (1998) 877–882.
[8] Y. Zhu, T. Tan, Y. Wang, Font recognition based on global texture analysis, IEEE Trans. PAMI 23 (10) (2001) 1192–1200.
[9] X. Ding, L. Chen, T. Wu, Character independent font recognition on a single Chinese character, IEEE Trans. PAMI 29 (2) (2007) 195–204.
[10] D. Tao, L. Jin, S. Zhang, et al., Sparse discriminative information preservation for Chinese character font categorization, Neurocomputing 129 (2014) 159–167.
[11] S. Zhang, L. Jin, D. Tao, et al., A faster method for Chinese font recognition based on Harris Corner, IEEE Trans. Syst. Man Cyber. (2013) 4271–4275.
[12] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: Proceedings of NIPS, 2012, pp. 1097–1105.
[13] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: Proceedings of ECCV, 2014, pp. 818–833.
[14] K. Simonyan, A. Zisserman, 2015. Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv: 1409.1556.
[15] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of CVPR, 2016, pp. 2818–2826.
[16] X. Bai, B. Shi, C. Zhang, X. Cai, L. Qi, Text/non-text image classification in the wild with convolutional neural networks, Pattern Recognit. (2017) 437–466.
[17] C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, 2016. Inception-v4, Inception-ResNet and the impact of residual connections on learning, arXiv preprint arXiv: 1602.07261.
[18] C. Szegedy, W. Liu, Y. Jia, et al., Going deeper with convolutions, in: Proceedings of CVPR, 2015, pp. 1–9.
[19] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of CVPR, 2016, pp. 770–778.
[20] K.M. He, X.Y. Zhang, S.Q. Ren, et al., Spatial pyramid pooling in deep convolutional networks for visual recognition, in: Proceedings of ECCV, 2014, pp. 346–361.
[21] B. Graham, Sparse arrays of signatures for online character recognition, 2013. arXiv preprint arXiv: 1308.0371.
[22] Z. Zhong, L. Jin, Z. Xie, High performance offline handwritten Chinese character recognition using GoogleNet and directional feature maps, in: Proceedings of ICDAR, 2015, pp. 846–850.
[23] D. Ciresan, U. Meier, J. Schmidhuber, Multi-column deep neural networks for image classification, in: Proceedings of CVPR, 2012, pp. 3642–3649.
[24] W. Yang, L. Jin, D. Tao, Z. Xie, Z. Feng, Dropsample: a new training method to enhance deep convolutional neural networks for large-scale unconstrained handwritten Chinese character recognition, Pattern Recognit. 58 (2016) 190–203.
[25] R. Girshick, J. Donahue, T. Darrell, et al., Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of CVPR, 2014, pp. 580–587.
[26] S. Ren, K. He, R. Girshick, et al., Faster R-CNN: Towards real-time object detection with region proposal networks, in: Proceedings of NIPS, 2015, pp. 91–99.
[27] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. PAMI 35 (8) (2013) 1798–1828.
[28] G.E. Hinton, N. Srivastava, A. Krizhevsky, et al., 2012. Improving neural networks by preventing co-adaptation of feature detectors, arXiv preprint arXiv: 1207.0580.
[29] L. Wan, M. Zeiler, S. Zhang, et al., Regularization of neural networks using dropconnect, in: Proceedings of ICML, 2013, pp. 1058–1066.
[30] G. Huang, Y. Sun, Z. Liu, D. Sedra, K.Q. Weinberger, Deep networks with stochastic depth, in: Proceedings of ECCV, 2016, pp. 646–661.
[31] F. Bastien, Y. Bengio, A. Bergeron, et al., 2010. Deep self-taught learning for handwritten character recognition, arXiv preprint arXiv:1009.3589.
[32] P.Y. Simard, D. Steinkraus, J.C. Platt, Best practices for convolutional neural networks applied to visual document analysis, in: Proceedings of ICDAR, 2003, p. 958.
[33] X. Frazão, L.A. Alexandre, DropAll: generalization of two convolutional neural network regularization methods, in: Proceedings of ICIAR, 2014, pp. 282–289.
[34] S. Gidaris, N. Komodakis, Object detection via a multi-region and semantic segmentation-aware CNN model, in: Proceedings of ICCV, 2015, pp. 1134–1142.
[35] S. Ioffe, C. Szegedy, 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv: 1502.03167 .
[36] M. Lin, Q. Chen, S. Yan, Network in network, International Conference on Learning Representations, 2014 arXiv preprint arXiv: 1312.4400.
[37] L. Ji, G. Wei, Handwritten Chinese character recognition with directional decomposition cellular features, J. Circuit Syst. Comput. 8 (04) (1998) 517–524.
[38] Y.-H. Tseng, C.-C. Kuo, H.-J. Lee, Speeding up Chinese character recognition in an automatic document reading system, Pattern Recognit. 31 (11) (1998) 1601–1612.
[39] J. Tsukumo, H. Tanaka, Classification of hand-printed Chinese characters using nonlinear normalization and correlation methods, in: Proceedings of ICPR, 1988, pp. 168–171.
[40] P. Sermanet, D. Eigen, X. Zhang, et al., 2013. Overfeat: Integrated recognition, localization and detection using convolutional networks, arXiv preprint arXiv: 1312.6229.
[41] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of ICML, 2010, pp. 807–814.
[42] Y. Jia, E. Shelhamer, J. Donahue, et al., Caffe: convolutional architecture for fast feature embedding, in: Proceedings of ACM MM, 2014, pp. 675–678.
[43] D. Tao, X. Lin, L. Jin, et al., Principal component 2-D long short-term memory for font recognition on single Chinese characters, IEEE Trans. Cybern. 46 (3) (2015) 756–765.
[44] A. Graves, J. Schmidhuber, Offline handwriting recognition with multidimensional recurrent neural networks, in: Proceedings of NIPS, 2008, pp. 545–552.
[45] A. Gupta, A. Vedaldi, A. Zisserman, Synthetic data for text localization in natural images, in: Proceedings of CVPR, 2016.
[46] P. Perez, M. Gangnet, A. Blake, Poisson image editing, ACM TOG 22 (3) (2003) 313–318.
[47] D. Song, D. Tao, Biologically inspired feature manifold for scene classification, IEEE Trans. Image Process. 19 (1) (2010) 174–184.
[48] D. Song, D. Tao, C1 units for scene classification, in: Pattern Recognition, Proceedings of ICPR, 2008, pp. 1–4.
[49] Q.M. Murala, Jonathan Wu, Expert content-based image retrieval system using robust local patterns, J. Visual Commun. Image Represent. 25 (6) (2014) 1324–1334.
[50] C.-C. Chang, Lin C.-J, LIBSVM: a library for support vector machines, ACM TIST 2 (3) (2011) 27.

Shuangping Huang received her M.S. and Ph.D. degrees from South China University of Technology in 2005 and 2011, respectively. She is now an associate professor at the School of Electronic and Information Engineering, South China University of Technology. Her research interests include machine learning, computer vision, document analysis, and scene text analysis.

Zhuoyao Zhong is pursuing Ph.D. in Information and Communication Engineering at South China University of Technology. He received his B.S. in Electronics and Information Engineering from South China University of Technology in 2015. His research interests include machine learning, pattern recognition, and text detection and recognition from natural scenes.

Lianwen Jin received his B.S. degree from the University of Science and Technology of China, Anhui, China, and Ph.D. degree from the South China University of Technology, Guangzhou, China, in 1991 and 1996, respectively. He is now a professor at the College of Electronic and Information Engineering, South China University of Technology. He is the author of more than 150 scientific papers, and has received the MOE award from the New Century Excellent Talent Program in 2006 and the Guangdong Pearl River Distinguished Professor Award in 2011. His current research interests include image processing, machine learning, pattern recognition, computer vision, and intelligent systems.



Shuye Zhang received his B.S. degree from Sun Yat-sen University, Guangzhou, China. He is currently pursuing Ph.D. in Information and Communication Engineering at South China University of Technology, Guangzhou, China. His current research interests include computer vision and pattern recognition.

Haobin Wang is pursuing his Masters in Signal and Information Processing at South China University of Technology. He received his BS in Electronic and Information Engineering from South China University of Technology in 2016. His research interests include machine learning, pattern recognition, and scene text detection and recognition.