Received March 26, 2019, accepted April 9, 2019, date of publication April 18, 2019, date of current version May 1, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2911964

Integrating Local CNN and Global CNN for Script Identification in Natural Scene Images

LIQIONG LU[1], YAOHUA YI[1], FALIANG HUANG[2], KAILI WANG[1], AND QI WANG[3]
[1]School of Printing and Packaging, Wuhan University, Wuhan 430072, China
[2]College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350007, China
[3]School of Light Industry and Food Engineering, Nanjing Forestry University, Nanjing 210037, China

Corresponding author: Yaohua Yi (whudcil@whu.edu.cn)

This work was supported in part by the Key Laboratory of Satellite Mapping Technology and Application and National Administration of Surveying, Mapping and Geoinformation under Grant KLSMTA-201702 and Grant KLSMTA-201604, in part by the National Science and Technology Major Project under Grant 2017ZX01030102, and in part by the Research Fund of Guangxi Key Lab of Multi-source Information Mining and Security under Grant MIMS17-01.

ABSTRACT Script identification in natural scene images is a key pre-step for text recognition and is also an indispensable condition for automatic text understanding systems that are designed for multi-language environments. In this paper, we present a novel framework integrating Local CNN and Global CNN both of which are based on ResNet-20 for script identification. We first obtain a lot of patches and segmented images based on the aspect ratios of the images. Subsequently, these patches and segmented images are used as inputs to Local CNN and Global CNN for training, respectively. Finally, to get the final results, the Adaboost algorithm is used to combine the results of Local CNN and Global CNN for decision-level fusion. Benefiting from such a strategy, Local CNN fully exploits the local features of the image, effectively revealing subtle differences among the scripts that are difficult to distinguish such as English, Greek, and Russian. Moreover, Global CNN mines the global features of the image to improve the accuracy of script identification. The experimental results demonstrate that our approach has a good performance on four public datasets.

INDEX TERMS Script identification, Local CNN, Global CNN, ResNet-20, decision-level fusion.

I. INTRODUCTION

Script identification is an inevitable pre-step of natural scene text understanding under multi-lingual scenarios. There are two steps in script identification for scene text images: one is text localization to get pre-segmented text lines [1]–[4], and the other is identifying the script types of these pre-segmented text lines. This paper focuses on the second step: identifying the script types of text in natural scene images. It is also viewed as an image classification problem. Specifically, we will classify the pre-segmented text lines that have been located in Figure 1 into specific script types.

Script identification has achieved very good results in document analysis [5], [6] and video analysis [7], [8]. In contrast to the text in document images and video images, text in natural scene images has many additional challenges, such as more complex backgrounds, less contextual information and large variations in fonts, colors or layout shapes. Based on the above reasons, the methods of script identification in document analysis or video analysis are not suitable for natural scene images. Recently, Convolutional neural networks (CNNs) have become the dominant machine learning

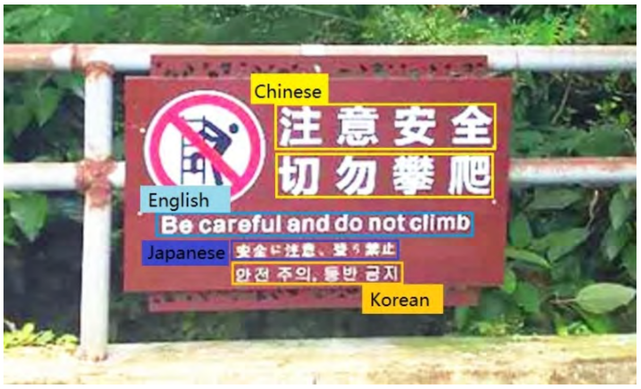


FIGURE 1. Script identification in a natural scene text image.

The associate editor coordinating the review of this manuscript and approving it for publication was Long Wang.

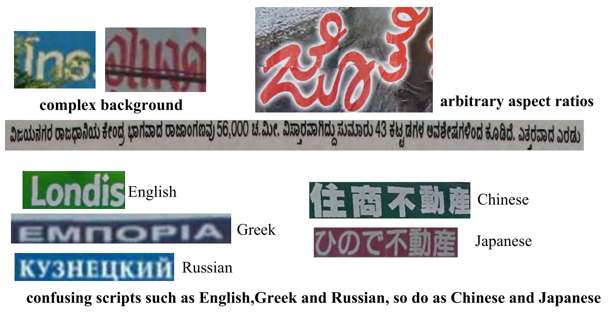

complex background
arbitrary aspect ratios
English
Greek
Russian
Chinese
Japanese
confusing scripts such as English,Greek and Russian, so do as Chinese and Japanese

FIGURE 2. Pre-segmented text lines in SIW-13 concerning the challenges for script identification.

approach for visual tasks and CNNs are state-of-the-art models for many image classification tasks because of their strong capacity and invariance to translation and distortions [9], [10]. However these classification methods based on CNN are not suitable to be directly applied for script identification because of two bigger challenges. One challenge is that the extremely variable aspect ratios of scene text images bring much difficulty to any CNN classifier with a fixed size image as input. The other challenge is that there are many parts that are similar or even nearly identical in different script images. Those confusing scripts, such as English, Greek and Russian, as well as Chinese and Japanese, having similar appearances and even sharing the same characters, make script identification more difficult. There are some pre-segmented text lines in SIW-13 [11] concerning the above two challenges for script identification in Figure 2.

The existing outstanding methods use CNNs to mine discriminative features to accurately identify those confusing scripts [11]–[13]. These methods using patches as inputs to CNNs are good at mining the local features of scene text images and coping with the two challenges that were mentioned earlier. However dividing a scene image into patches loses some global features and degrades the overall identification performance. The common method integrating the local and global features of a scene text image is using one CNN to get the local and global features from different layers, and then combining them together via up-sampling or down-sampling. Considering variable special aspect ratios, with which the content of images may be warped as inappropriate inputs, and indeterminable weights of local and global features, we argue that this operation is not suitable for script identification which is confirmed by the experiments in a subsequent subsection.

In this work, we integrate Local CNN and Global CNN to fully exploit the local features and global features of images for script identification. Inspired by the work in [12], [14], Local CNN is designed to use patches with the same size as inputs and an ensemble of N networks based on ResNet-20 [9] as the architecture. In order to highlight the first network that is used in the testing stage, we use a higher-accuracy model to fine-tune the first network and a relatively low-accuracy model to fine-tune the other networks in Local CNN. Then accuracy of the first network becomes increasingly higher than the other networks as the training progresses. Through this fine-tuning strategy, Local CNN fully exploits the local features of images, effectively revealing the subtle differences among the scripts that are difficultly to distinguish such as English, Greek and Russian, as well as Chinese and Japanese. In addition, using patches as input avoids the problems that are caused by the extremely variable aspect ratios of scene text images. Local CNN in our approach is similar to ECN in [12], but there are two differences between them: different network architectures and different fine-tuning strategies. Actually, Local CNN has achieved better classification accuracy than ECN, which was verified in the experiments. Global CNN uses simple segmented images as inputs, and its architecture is based on a single ResNet-20. It mines the global features of images to improve the accuracy of script identification. In particular,, our method of segmenting images can retain the global features of image, and avoid the problems that are caused by the extremely variable aspect ratios of scene text images. Finally, the Adaboost [15] algorithm is used to combine the results of Local CNN and Global CNN for decision-level fusion to obtain the final results. The experimental results on four public datasets named SIW-13 [11], MLe2e [12], CVSI-2015 [16] and ICDAR-2017 [17] demonstrate the good performance of our approach.

II. RELATED WORK

Previous work on script identification mainly focused on texts in documents [5], [18] and videos [19], [20]. In the field of document analysis, the methods of script identification mainly belonged to two broad categories: structure-based and visual appearance-based techniques. The general procedure first extracted the features and then used different classifiers such as SVM, KNN etc. to obtain the classified results for several scripts. In 2015, the ICDAR Competition on Video Script identification (CVSI) was held [16]. Unlike the traditional approaches based on hand-crafted features, the top performing methods in the competition were all based on CNN. The competition winner (Google) was also based on CNN, but it applied a binarization pre-processing step to the input images. This pre-processing is not suitable for natural scene images with complex backgrounds.

The existing methods of script identification in natural scene images can be divided into two categories. The first type uses CNN to automatically mine script features, and the second manually designs different features for script identification. Overall, the CNN based methods have better performance than hand-crafted features based methods [12].

The first study of script identification in natural scene images was conducted by Shi et al. [21]. The authors proposed a CNN based framework named Multi-stage Spatially-sensitive Pooling Network (MSPN) to cope with the extremely variable aspect ratios of scene text images. They extended their work in [11] in which the mid-level representations were integrated into a single CNN for script

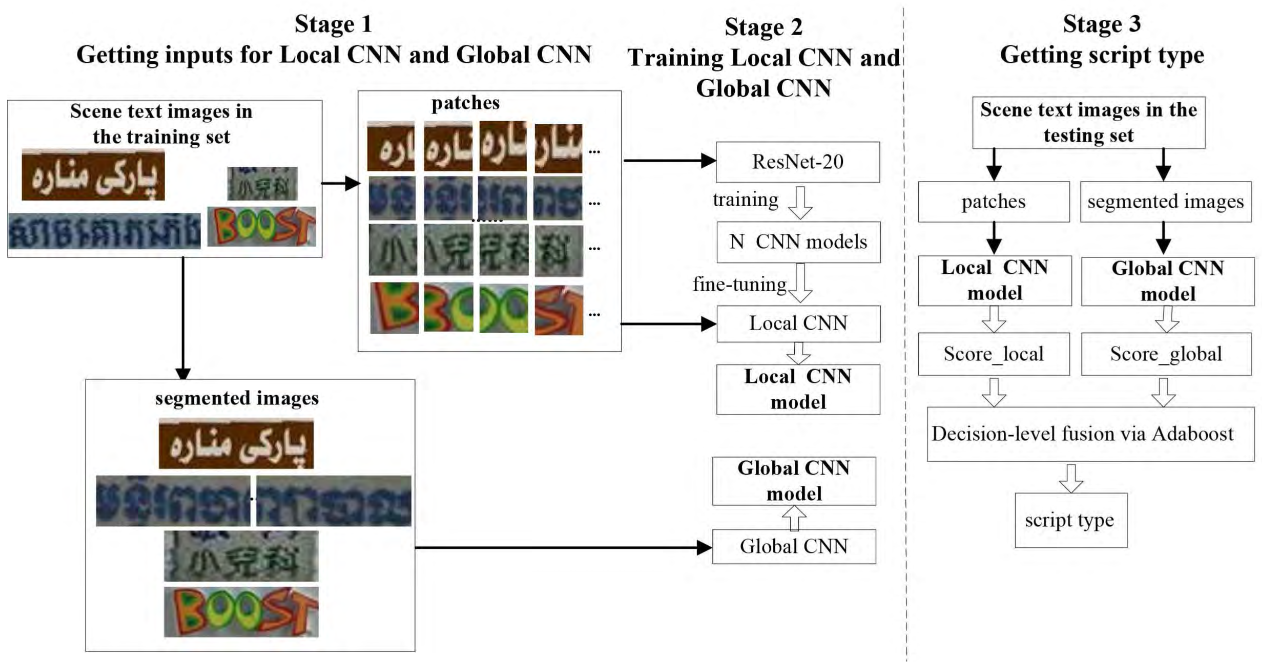


FIGURE 3. **The overall illustration of our approach. Stage1 and Stage 2 belong to the training stage and Stage 3 belongs to the testing stage.**

identification. Jieru Mei et al. [22] exploited two important factors named the image representation and the spatial dependencies within text line, then brought them and a Convolutional Neural together into one end-to-end trainable network to identify scripts of images. At the same time, they adopted an average pooling structure to deal with input images with arbitrary sizes. In [23], Gomez et al. combined the convolutional features, the Naive-Bayes Nearest Neighbor classifier and a simple weighting strategy to discover the most discriminative patches per class. They extended their work in [12] in two ways: making use of a much deeper Convolutional Neural Network model and replacing the NBNN with a patch-based classification rule that can be integrated in the CNN training process by using an Ensemble of Conjoined Networks. In [13], Ankan Kumar Bhunia et al. proposed a novel method that involved extraction of the local and global features using CNN-LSTM frame work and attention-based patch weights calculated by applying softmax layer after LSTM. Like the above approaches, our approach is also based on CNN. The methods in [12], [13] both used patches as inputs to CNN for script identification. Our method uses patches and segmented images as inputs to CNN for script identification.

There were some methods of script identification for natural scene images that were based on hand-crafted features. Firstly, different hand-crafted features were extracted, and then various classifiers such as SVM, Random Forest and so on were trained to predict the script types in these methods [24]–[27]. These features included Local Binary Pattern (LBP), Center Symmetric Local Binary Pattern (CSLBP), Directional Local Extrema Pattern (DLEP), the Gabor feature, the Log-Gabor feature, and the wavelet feature and so on.

III. THE PROPOSED APPROACH

A. OVERVIEW

Given an image I, we predict its script class $c \in \{1, 2, \ldots, C\}$ where 1 to C denotes a specific script type. The overall illustration of our approach is shown in Figure 3. In the training stage, which is Stage 1 and Stage 2 in the illustration, we first extract patches from the scene text images and use these patches as inputs to train Local CNN, and then we obtain segmented images based on the aspect ratio and use them as inputs to train Global CNN. In the testing stage, which is Stage 3 in the illustration, we use Local CNN model and Global CNN model to obtain two prediction scores. Finally, the Adaboost algorithm is used to conduct the decision-level fusion of the results of Local CNN and Global CNN.

B. INPUTS OF LOCAL CNN AND GLOBAL CNN

Local CNN is designed to use patches as inputs. Therefore we first densely extract patches with the same size from images. In the same way as in [12], we set the size of each patch to 32×32. For an image, we first convert it to a gray-scale image and resize the height to 40 pixels while keeping the same aspect ratio as the original image. Then, we densely extract patches with a size of 32×32 from the horizontal and vertical directions of the resized image. The step size of the extraction is 8 pixels. The entire process of extracting patches is automatically performed by the program without human intervention.

For the traditional CNN, the size of the input images must be the same. However, because the aspect ratios of text images vary greatly, if the images are directly resized to the same size, it will cause an overall distortion and affect the identification accuracy. At this point, Global CNN is designed to use segmented images as input. By observing the text lines in natural scene images, we find that when the width is three times the height, the global features of the text line are basically maintained. Therefore we set the size of each segmented image to 40×120. First, we convert an image to a gray-scale image and then resize the height to 40 pixels while keeping the same aspect ratio as the original image. After resizing, if the width is less than 240 pixels, we directly resize the image to make the width 120 pixels. If the width is greater or equal to 240 pixels, we split the image into multiple sub-images with the same size of 40×120. The detailed segmentation process is shown in Algorithm 1. We show all the patches and segmented images that are acquired by a scene text image in Figure 4.

Algorithm 1 Image Segmentation Process

```
Input:  scene text images
Output:  segmented images
 1: for each $i \in [1, I]$ do
 2:     resizing image $i$ to keep the height 40 pixels
 3:     $D_i$ = the width of resized image
 4:     if $D_i < 240$ then
 5:         $segImg(i, 1)$ = resizing image to [40,120];
 6:         $splitNum_i = 1$;
 7:     else
 8:         $splitNum_i = D_i/120$;
 9:         $iFirstPos = 0$;
10:         for each $j \in [1, splitNumi - 1]$ do
11:             $segImg(i, j)$ = imcrop($iFirstPos$,0);
12:             $iFirstPos = iFirstPos + 120$;
13:         end for
14:         the last sub image = imcrop($iFirstPos$, 0);
15:         $segImg(i, splitNum_i)$ = resizing image to [40,120];
16:     end if
17: end for
18: return  $segImg(i, j)_{i=1toI, j=1toSplitNum_i}$
```

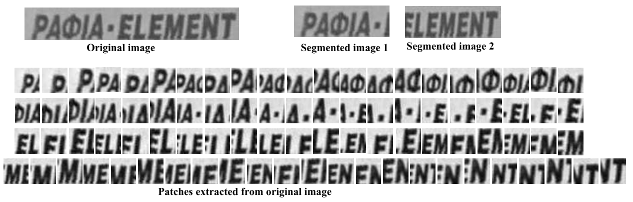


FIGURE 4. All segmented images and patches acquired by a scene text image.

It is worth mentioning that the text in the images in the ICDAR-2017 dataset is not all horizontal, and there is some vertical text or text that is inclined at a certain angle. Therefore, for the images in the ICDAR-2017 datasets, we conduct a special operation. If the height of the image is larger than the width, we will rotate the image by 90 degrees before extracting the patches or segmenting images.

C. LOCAL CNN

1) ARCHITECTURE

Local CNN is designed as an ensemble of N networks whose basic architecture is ResNet-20. ResNet is a very good structure for CNN and has achieved surprising performance in the field of image classification [9]. ResNet-20 is a 20-layer ResNet network that is very suitable for the classification task in small images. In Local CNN, N networks are joined at their outputs with an "Eltwise" layer, and a "Softmax" layer is behind the "Eltwise" layer to get the last script type. The architecture of Local CNN is shown on the left side in Figure 5. There are N patches that are extracted from the same scene text image as inputs to Local CNN.

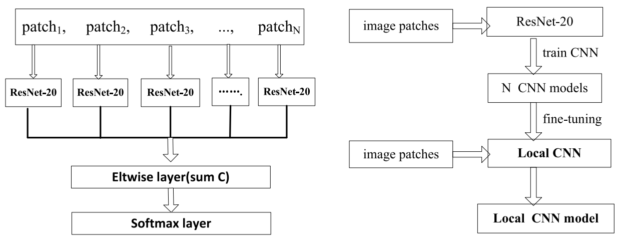


FIGURE 5. The left side is the architecture of Local CNN and the right side is the illustration for training Local CNN.

We use $(x_1, x_2 \ldots x_N)$ to denote the inputs of Local CNN. The output of the j^{th} network in Local CNN is a vector of length C that is denoted by $(Score_{j1}, Score_{j2} \ldots Score_{jC})$ and $Score_{ji}$ is the evaluation score of script type i. The response of the "Eltwise" layer is the sum of the outputs of the N networks and the details are shown in formula (1). In formula (1), $(y_1, y_2 \ldots y_C)$ is the output of the "Eltwise" layer. The response of the "Softmax" layer is shown in formula (2) and $(p_1, p_2 \ldots p_C)$ is a vector consisting of the probabilities of all script types. The script type with the highest probability is the last classification result in formula (2). We use the Cross Entropy loss to define the loss function. It is displayed in equation (3), where p_i denotes the predicted probability of the script type of sample i and M denotes the number of samples.

$$\begin{pmatrix} y_1 \\ \ldots \\ y_C \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{N} Score_{j1} \\ \ldots \\ \sum_{j=1}^{N} Score_{jC} \end{pmatrix} \quad (1)$$

$$where\ (Score_{j1}, Score_{j2} \ldots Score_{jC}) = ResNet_j(x_j)$$

$$\begin{pmatrix} p_1 \\ \ldots \\ p_C \end{pmatrix} = \begin{pmatrix} exp(y_1)/\sum_{i=1}^{C} exp(y_i) \\ \ldots \\ exp(y_C)/\sum_{i=1}^{C} exp(y_i) \end{pmatrix} \quad (2)$$

$$loss = -\sum_{i=1}^{M} log(p_i) \quad (3)$$

2) FINE-TUNING STRATEGY

The purpose of using an ensemble of multiple networks is to highlight the first network that is used to predict the script type during the testing stage. We analyze the architecture of Local CNN. In the "Eltwise" layer, the sum of the outputs of the N networks is used to predict the script type. It means that all networks in Local CNN contribute to the final result. We believe that using the trained models of ResNet-20 to fine-tune Local CNN can improve the accuracy. Therefore, like [12], only using a trained CNN model to fine-tune the first network is a good choice. However, it is not good for the final accuracy because of the low accuracy of the other networks. We use a compromise proposal to balance the first network and others in Local CNN. We design a fine-tuning strategy that will ensure that the first network has higher performance and that the other networks have little impact on the final result.

For an image I, its script label is S_i. We extract M patches from image I, each patch's corresponding script label is also S_i. We first use all patches that are extracted from scene images as inputs to train ResNet-20 to obtain N CNN models that are generated by different iterations of training. Then, we use these N CNN models to fine-tune Local CNN. In general, the CNN models of the different iterations have different levels of accuracy and CNN models with higher iterations always have higher accuracy. To highlight the first network, we use the CNN model with higher accuracy to fine-tune the first network and use the CNN models with relatively lower accuracy to fine-tune the remaining $N-1$ networks in Local CNN. For SIW-13 dataset, we use the CNN model that is obtained from the 110,000 iterations to fine-tune the first network and the CNN model that is obtained from the 40,000 iterations to fine-tune the remaining $N-1$ networks.

When training Local CNN, we select N patches that are randomly extracted from the same image as the inputs, and this image's corresponding script type is used as the script label. Using this strategy, the accuracy of the first network during training will be increasingly higher than other networks, and then we get Local CNN model that is used in the testing stage. The illustration for training Local CNN is shown on the right side in Figure 5.

D. GLOBAL CNN

Local CNN is a patch based CNN and dividing the scene text image into patches loses the global features of the images. Here Global CNN is a powerful complement to Local CNN and is designed to use the whole image or a simple segmented image as input. Global CNN is good for mining the global features of images and helps to improve the script identification performance. For Global CNN, the size of the input image is 40×120. Therefore, we use a single ResNet-20 (including a "Softmax' layer) as the architecture. The training of Global CNN is too simple to be introduced in detail. After the training of Global CNN, we obtain Global CNN model.

It is worth mentioning that the training processes of Local CNN and Global CNN are independent. So in the training stage we can train Local CNN and Global CNN in parallel and then combine the results in the testing stage.

E. INFERENCE

In the testing stage, we first obtain the patches and segmented images in the testing set. For an image I in the testing set, we extracted M patches that are denoted as $(Patch_1, Patch_2 \dots Patch_M)$. Supposing that the segmented images of I are denoted as $(Split_1, Split_2 \dots Split_S)$,$M$ and S are the numbers of patches and simple segmented images respectively. If image I does not need to be split, S is 1 and $Split_1$ is the resized image I. When the input is $Patch_k$ or $Split_k$, the output of the last fully connected layer (''FC_20'') in our case of Local CNN model or Global CNN model is a vector of length C, which is denoted as $(Score\_Local\_CNN_{k1}, \dots, Score\_Local\_CNN_{kC})$ *or* $(Score\_Global\_CNN_{k1}, \dots, Score\_Global\_CNN_{kC})$, respectively. C is the number of script types in the dataset. $Score\_Local\_CNN_{ki}$ or $Score\_Global\_CNN_{ki}$ corresponds to the evaluation score for script type i that is obtained by Local CNN model or Global CNN model, respectively. We obtain $Score\_local$ and $Score\_global$ using formulas (4) and (5). In formulas (4) and (5), $Local\_CNN_{FC\_20}$ and $Global\_CNN_{FC\_20}$ denote the response functions of the last fully connected layer of Local CNN model and Global CNN model, respectively.

$$\begin{pmatrix} Score\_local_1 \\ \dots \\ Score\_local_C \end{pmatrix} = \begin{pmatrix} \frac{1}{M}\sum_{k=1}^{M} Score\_Local\_CNN_{k1} \\ \dots \\ \frac{1}{M}\sum_{k=1}^{M} Score\_Local\_CNN_{kC} \end{pmatrix} \tag{4}$$

where $(Score\_Local\_CNN_{k1}, \dots, Score\_Local\_CNN_{kC}) = Local\_CNN_{FC\_20}(Patch_k)$

$$\begin{pmatrix} Score\_global_1 \\ \dots \\ Score\_global_C \end{pmatrix} = \begin{pmatrix} \frac{1}{S}\sum_{k=1}^{S} Score\_Global\_CNN_{k1} \\ \dots \\ \frac{1}{S}\sum_{k=1}^{S} Score\_Global\_CNN_{kC} \end{pmatrix} \tag{5}$$

where $(Score\_Global\_CNN_{k1}, \dots, Score\_Global\_CNN_{kC}) = Global\_CNN_{FC\_20}(Split_k)$

After obtaining the values of $Score\_local$ and $Score\_global$, the Adaboost algorithm is used to conduct the decision-level fusion of the results of Local CNN and Global CNN to get the last script type. Adaboost is an iterative algorithm whose core idea is to train different classifiers (weak classifiers) for the same training set, and then combine these weak classifiers to form a stronger final classifier (strong classifier). It is a widely used decision-level fusion method. Here, we use the Adaboost algorithm to solve our specific problem. Assume that there are N training samples (x_i, y_i) where $i = 1, 2, \dots, N$, x_i denotes the sample (patch or segmented image), y_i is the label corresponding to the script type and $y\epsilon(Y = \{1, 2, .., C\})$. The weight vector W is an $N \times C$

Algorithm 2 Decision-Level Fusion of Local and Global CNN

```
Input: $(x_i, y_i)_{i=1,2,...,N}$, $Score\_local$, $Score\_global$, $T$
Output: $weight\_local$, $weight\_global$
 1: $D_1(i) = 1/N(i = 1, 2, \ldots, N)$
 2: $W^1_{i,y} = D_1(i)$;
 3: $t = 0$;
 4: while $t < T$ do
 5:     $W^t_i = \sum_{y \neq y_i} W^{t-1}_{i,y}$; $D^t_i = w^t_i / \sum^N_{i=1} w^t_i$;
 6:     //$Q(i, y)$ is a weighting function
 7:     //and $\sum_{y \neq y_i} Q(i, y) = 1$
 8:     $Q_t(i, y) = W^t_{i,y} / W^t_i (y \neq y_i)$;
 9:     //Using D and Q to calculate the error of $h_t$,
10:     //$h_t$ is the confidence of classifying $x$ as $y$
11:     if $(mod(t, 2) == 0)$ then
12:         $h_t = softmax(Score\_local)$;
13:     else
14:         $h_t = softmax(Score\_global)$;
15:     end if
16:     $error_t = 0.5 * (\sum^N_{i=1} D_t(i)[1 - h_t(x_i, y_i) + \sum_{y \neq y_i} Q_t(i, y) h_t(x_i, y)])$;
17:     //Update the weight vector
18:     $\beta_t = error_t/(1 - error_t)$;
19:     $W^{t+1}_{i,y} = W^t_{i,y} \beta_t^{0.5*(1-h_t(x_i,y_i)-h_t(x_i,y))}$;
20:     t = t + 1;
21: end while
22: $weight\_local = \sum_{mod(t,2)=0} lgâĄą(1/\beta_t)$;
23: $weight\_global = \sum_{mod(t,2)\neq 0} lgâĄą(1/\beta_t)$;
24: return $weight\_local$, $weight\_global$;
```

vector that denotes the sample weights of the different script types. T is the number of iterations. The detailed process is shown in Algorithm 2.

$$\begin{pmatrix} y_1 \\ \ldots \\ y_C \end{pmatrix} = \begin{pmatrix} weight\_local * Score\_local_1 \\ +weight\_global * Score\_global_1 \\ \ldots \\ weight\_local * Score\_local_C \\ +weight\_global * Score\_global_C \end{pmatrix} \quad (6)$$

$$script\_type = type\_of\_max(y_1, y_2 \ldots y_C) \quad (7)$$

With algorithm 2, we obtain the values of $weight\_local$ and $weight\_global$. Then we get the last predict script type through formulas (6) and (7). According to formula (7), the maximum value of $y_1, y_2 \ldots y_C$ is found and then the corresponding script type is the final predicted result.

IV. EXPRIMENTS

We verify our approach and compare the results with those of other approaches on four public datasets named SIW-13, MLe2e, CVSI-2015 and ICDAR-2017. There are all natural scene text images in SIW-13, MLe2e and ICDAR-2017 datasets. CVSI-2015 mainly contains overlay video text with a few instances of scene text images. In SIW-13 dataset, there are 16,291 pre-segmented text lines including 13 scripts: Arabic, Cambodian, Chinese, English, Greek, Hebrew, Japanese, Kannada, Korean, Mongolian, Russian, Thai and Tibetan. In MLe2e dataset, there are 4 scripts including Latin, Chinese, Kannada and Hangul, and 1177 and 642 pre-segmented text lines are in the training set and the testing set respectively. In CVSI-2015, there are 10 scripts including Arabic, English, Hindi, Bengali, Oriya, Gujrathi, Punjabi, Kannada, Tamil, and Telegu with approximately 1100 pre-segmented text lines in each script. The ICDAR-2017 dataset has 68,613 pre-segmented text word images for training. The validation set has 16,255 word images. The dataset consists of 9 languages including Arabic, English, French, Chinese, German, Korean, Japanese, Italian, and Bangla. Out of the above languages English, French, German and Italian share the same Latin script. Therefore, in our current work, these scripts are assigned the same script class: Latin. Additionally, isolated punctuation or other special characters are considered as a special script class, namely, Symbols. Hence, we have a total of 7 script classes.

A. IMPLEMENTATION DETAILS

We have used the open source Caffe [28] framework to run the deep learning running on commodity GPUs. All CNNs used in our approach are optimized using the stochastic gradient descent (SGD). When training the single ResNet-20, the initial learning rate is set to 10^{-2} and decreased by a factor of 10 after every 100,000 iterations. When training Local CNN, the initial learning rate is set to 10^{-3} and decreased by a factor of 10 after every 10,000 iterations. The momentum is set to 0.9 and the batch size is set to 64. For SIW-13 dataset, in the same way as in [12], the value of N is set to 10. The number of iterations of the first CNN model is 110,000 and the number of iterations of the remaining $N - 1$ CNN models is 40,000 in the training stage. All parameters will be discussed in subsequent subsections. The detailed architecture of ResNet-20 is shown in Table 1.

TABLE 1. The architecture of ResNet-20 that are used in our approach.

Layer Name	Output Size	ResNet-20
Input	32×32 or 40×120	
conv1_x	32×32 or 40×120	[3×3, 16] ×7
conv2_x	16×16 or 20×60	[3×3, 32] ×6
conv3_x	8×8 or 10×30	[3×3, 64] ×7
average_pool_fc	C	average pool + fc

B. SCRIPT IDENTIFICATION IN PRE-SEGMENTED TEXT LINES

The overall classification accuracy of our approach and the comparison with other approaches on the four public datasets are shown in Table 2. For all methods, the images in the same training set are used for training and the images in the same testing set are used to compare the overall classification performances. As shown in Table 2, our approach has good performance on the four datasets. Compared with the methods in [11]–[13], [21]–[23] which are all designed for

TABLE 2. Overall classification performance comparison with other methods on four datasets: MLe2e, SIW-13, CVSI-2015 and ICDAR-2017.

Method	MLe2e	SIW-13	ICDAR-2017	CVSI-2015
Ours	0.958	0.961	**0.932**	**0.983**
Ankan et al.[13]	**0.967**	**0.965**	0.902	0.977
ECN[12]	0.944	0.948	0.871	0.972
Mei et al.[22]	-	0.928	-	0.942
Hust[21]	-	0.880	-	0.967
Shi et al[11]	-	0.894	-	0.943
Gomez et al.[23]	0.911	0.769	-	0.979
Google[16]	-	-	-	**0.989**
Nicolaou et al.[27]	-	0.837	-	0.982
Baseline Sequence-based CNN[29](Early fusion)	0.898	0.889	-	0.936
Baseline SIFT[30] + Fisher Vectors + SVM	0.886	0.907	-	0.941
Baseline SIFT[30] +VLAD + SVM	0.902	0.892	-	0.939
Baseline SIFT[30] + Bag of Words +SVM	0.865	0.834	-	0.844

script identification in natural scene images, our approach has the highest overall accuracy on CVSI-2015 and ICDAR-2017 datasets and a slightly lower accuracy than the method in [13] on SIW-13 and MLe2e datasets.

There are two main differences between our method and the method in [13]. The first is the different input strategies. We use both patches and segmented images as inputs to CNN, while [13] only uses patches as inputs. The second is the different fusion strategies. The method in [13] utilizes the attention mechanism to conduct the feature-level fusion of multiple patches. Our method uses an ensemble learning technique. Specifically, the Adaboost algorithm is adapted to conduct the decision-level fusion of the results of Local CNN and Global CNN. After further investigation into the experimental datasets, we find that some non-horizontal texts exist in the ICDAR 2017 dataset, while in the other three datasets all texts are horizontal. For non-horizontal text, the approach in [13] extracts patches horizontally or vertically, which easily introduces more background areas into the patch and, thus may lead to low performance. Our approach uses Global CNN with segmented images as the input, which cooperates with Local CNN to handle these non-horizontal texts, which improves the accuracy by 3% on the ICDAR-2017 dataset. Our approach has better performance on ICDAR-2017 and CVSI-2015 datasets, because our Global CNN complements Local CNN, effectively improving the overall identification performance. We analyze the method in [13] on MLe2e and SIW-13 datasets, since its performance is slightly better than ours because of its effective weight distribution mechanism. In addition, the fact that there are only horizontal text images in SIW-13 and MLe2e datasets also highlights the weak advantage of the method in [13] over ours.

Google [16] has the highest accuracy of 0.989 on CVSI-2015 dataset. This method applied a binarization pre-processing step to the input images. This pre-processing is not suitable for scene images with complex backgrounds. Our approach is compared with other CNN-based approaches [27], [29], which are all designed for texts in videos. These approaches have good performance on CVSI-2015 dataset but have much lower performance than our approach on SIW-13 and MLe2e datasets. The main reason is that the text in a scene image has a much more complicated background than the text in a video. These approaches are not suitable for scene text script identification. We also compare our approach with the approaches based on hand-crafted features. We combine the Scale Invariant Feature (SIFT) [30] with three different encodings (Fisher Vectors, Vector of Locally Aggregated Descriptors (VLAD), and Bag of Words (BoW)) and the SVM classifier [12]. The results show that the overall accuracy of our approach is also higher than these approaches using hand-crafted features.

TABLE 3. The accuracy of each script on SIW-13 and comparison with other methods.

Script	Shi et al.[11]	Mei et al.[22]	ECN[12]	Ankan et al.[13]	Ours
Arabic	0.980	0.962	0.986	**0.990**	0.978
Cambodian	0.880	0.934	0.956	**0.990**	0.974
Chinese	0.880	0.940	0.930	0.920	**0.964**
English	0.710	0.836	0.884	0.980	0.892
Greek	0.810	0.894	0.896	**1**	0.930
Hebrew	0.910	0.938	0.934	**0.990**	0.970
Japanese	0.900	0.918	0.954	**0.970**	0.967
Kannada	0.910	0.918	0.960	0.920	**0.966**
Korean	0.950	0.956	0.978	0.930	**0.990**
Mongolian	0.960	0.970	**0.988**	0.980	0.984
Russian	0.790	0.870	0.876	**0.930**	0.902
Thai	0.940	0.936	0.978	0.950	**0.978**
Tibetan	0.970	0.986	0.998	0.970	**1**
Avg	0.894	0.928	0.948	**0.965**	0.961

There are 13 and 10 script types in SIW-13 and CVSI-2015 datasets, respectively. Therefore, we detail the accuracy of each script and compare the results with other approaches that are designed for scene text images in Table 3 and Table 4. From the results in Table 3, although the overall accuracy of our approach is slightly lower than that of the approach in [13], our approach has higher accuracy than the approach in [13] for Chinese, Kannada, Korean, Mongolian, Thai and Tibetan. The approach in [13] is a very good algorithm that considers the contributions of the different patches to the classification using different weights. We assign the same weight to each patch when adding scores. We think that the value of the score reflects the weight of the patch to a certain extent. Patches with higher scores contribute more to the final result. The good performance of our approach is also due to the integration of Local CNN and Global CNN.

We detail the accuracy of each script on CVSI-2015 dataset in Table 4. There are four tasks on CVSI-2015 dataset and in our experiments we only address Task-4, classifying all 10 scripts, as it is the most generic task. As shown in Table 5 we can see that our approach has good performance for scripts such as Bengali, Oriya, Gujrathi, Punjabi, Tamil and Telegu with relatively lower accuracy for Kannada.

We also list the confusion matrices of all four datasets in Figure 6. From the results of the confusion matrices,

TABLE 4. **The accuracy of each script on CVSI-2015 and compared with other approaches.**

Script	Shi et al.[11]	Hust[21]	ECN[12]	Ankan[13]	Ours
English	-	0.935	-	0.942	**0.982**
Hindi	-	0.963	-	0.965	**0.990**
Bengali	-	0.958	-	0.956	**1**
Oriya	-	0.985	-	0.983	**0.997**
Gujrathi	-	0.975	-	0.987	**0.997**
Punjabi	-	0.971	-	0.991	**1**
Kannada	-	0.927	-	0.986	0.86
Tamil	-	0.978	-	0.992	**1**
Telegu	-	0.978	-	0.977	**1**
Arabic	-	**1**	-	0.996	**1**
Avg	0.943	0.967	0.972	0.978	**0.983**

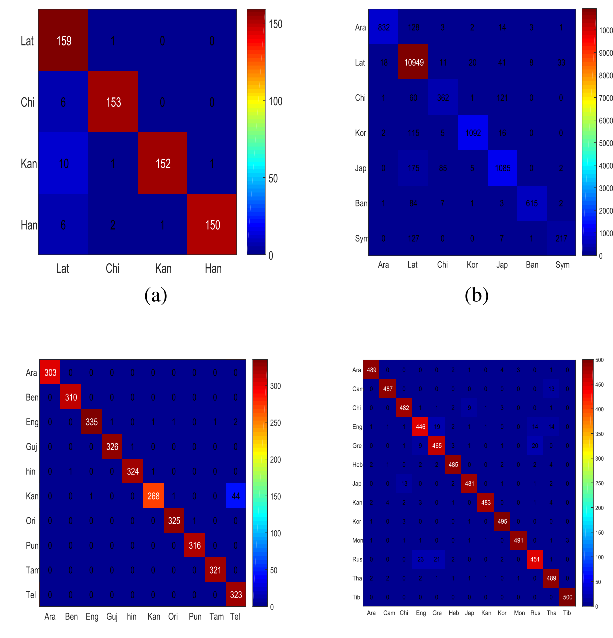

(a) (b) (c) (d)

FIGURE 6. **Confusion matrixes on four datasets. (a) MLe2e. (b) ICDAR-2017. (c) CVSI-2015. (d) SIW-13.**

we find that Kannada has a high probability of being misclassified as Telegu. Greek, Russian and English are often confused, as are Chinese and Japanese. These confusing scripts have very similar appearance and are also difficult to be distinguished by human eyes. In addition, each script has a certain probability of being misclassified as Latin in ICDAR-2017 dataset due to the imbalance of samples in the training set. There are 47,446 samples that are labeled as Latin, which is more than the sum of the samples (21,167) that are labeled as the other five scripts.

C. DISCUSSION

1) IMPACT OF DIFFERENT PARAMETERS

We set the value of the parameters using the experimental results from the validation set. For SIW-13, MLe2e and ICDAR-2017 datasets, we randomly choose 10% of the data from the training set as a validation set. In the training stage of Local CNN, N is set to 10 which is the same as in [12]. For SIW-13 dataset, the number of iterations that we select for the first CNN model is 110,000 because of its high accuracy. How we select the number of iterations for the remaining $N-1$ CNN models is very important. Our purpose is to highlight the first network in Local CNN, and so the number of iterations for the remaining $N-1$ CNN models must be less than 110,000. We set different values for the other $N-1$ CNN models and then compare the overall accuracy using the validation set. The accuracy of Local CNN is shown in Figure 7 where Ensemble-ResNet_110000_M denotes that the number of iterations for the first CNN model is 110,000 and the number of iterations for the remaining $N-1$ CNN models is M. From the results in Figure 7, Ensemble-ResNet_110000_40000 has the best accuracy of 0.955 using the validation set.

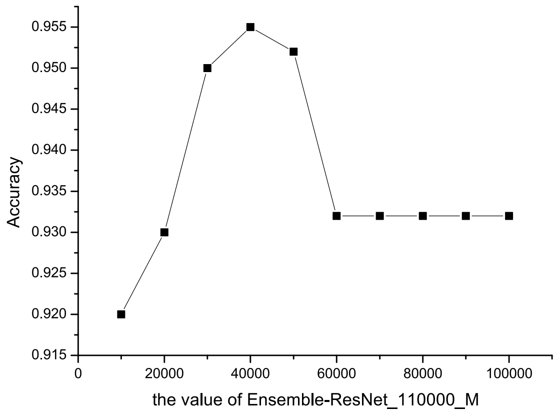


FIGURE 7. **The accuracy changes according to the value of *M*.**

TABLE 5. **The values of *weight_local* and *weight_global* on four datasets.**

Method	MLe2e	SIW-13	ICDAR-2017	CVSI-2015
weight_local	1.9043	1.008	1.650	0.511
weight_global	0.3393	0.1604	0.203	0.496

In the testing stage, we use the Adaboost algorithm to conduct the decision-level fusion of the results of Local CNN and Global CNN. We list the values of the fusion parameters in Table 5. From the results in Table 5 we can see that Local CNN has more contribution than Global CNN for SIW-13, MLe2e and ICDAR-2017 datasets. In addition, for CVSI-2015 dataset, Local CNN has almost the same contribution as Global CNN.

2) IMPACT OF LOCAL CNN AND GLOBAL CNN

In our approach, Local CNN and Global CNN are integrated for script identification. To better understand their effects, we have listed the classification accuracy of Local CNN and Global CNN on the four datasets in Table 6. Actually, the good performance is mainly due to the design of Local CNN. After Global CNN is integrated, the overall classification accuracy has been further improved.

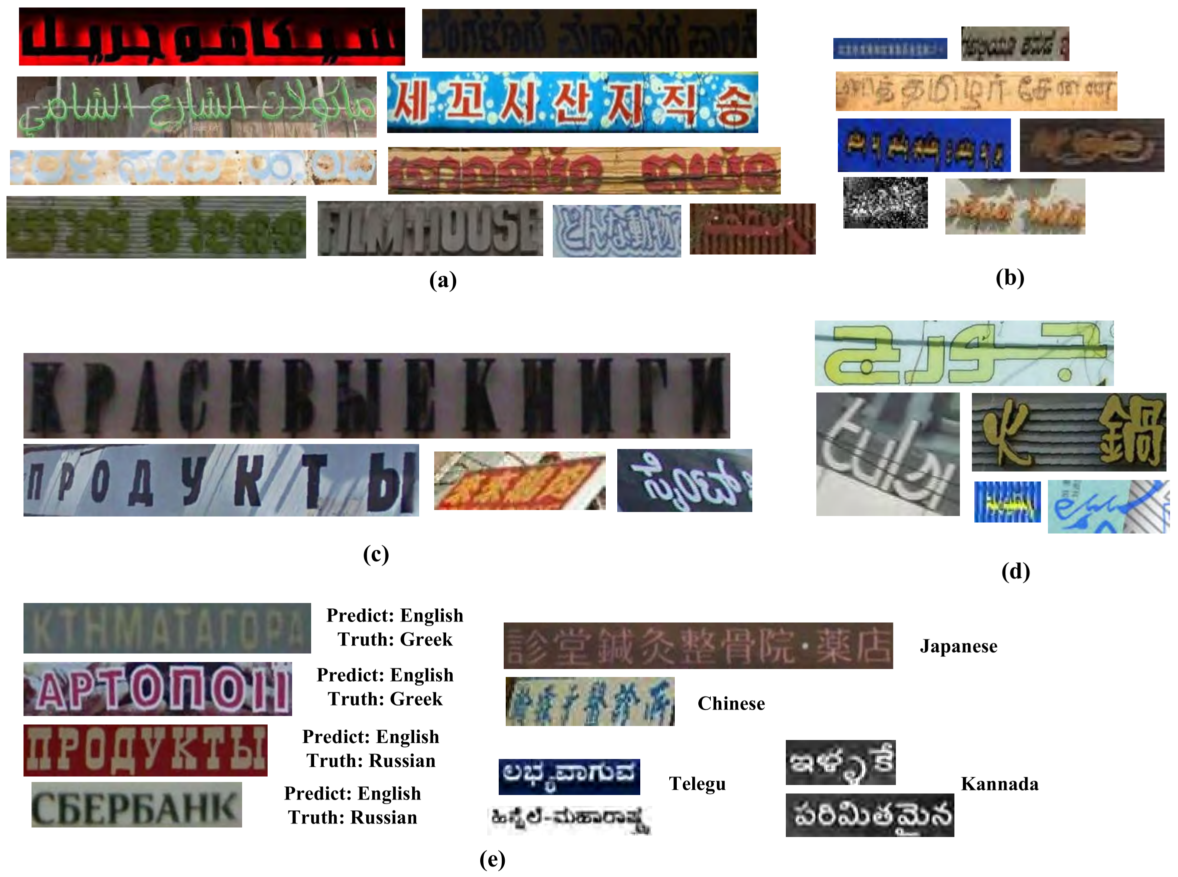


FIGURE 8. Samples that our approach fails to identify. (a) Complex background or background and text mixed together. (b) poor quality text. (c) Large text spacing or non-horizontal text. (d) shelters in front of text. (e) Great similar scripts such as Russian, Greek and English; Japanese and Chinese, Telegu and Kannada.

TABLE 6. Overall classification accuracy of Local CNN and Global CNN on four datasets.

Method	MLe2e	SIW-13	ICDAR-2017	CVSI-2015
Local CNN	0.957	0.953	0.917	0.971
Global CNN	0.900	0.915	0.914	0.953
Ours	**0.958**	**0.961**	**0.932**	**0.983**

TABLE 7. Overall classification accuracy of Global CNN with different methods of obtaining input images.

Method	MLe2e	SIW-13	ICDAR-2017	CVSI-2015
Global_CNN_Segment	**0.900**	**0.915**	**0.914**	**0.953**
Global_CNN_Resize	0.862	0.896	0.908	0.953

3) IMPACT OF SEGMENTATION OF IMAGES IN GLOBAL CNN

In our approach, we use segmented images as input to Global CNN, which preserves the global features of the image and avoids the problems that are caused by the extremely variable aspect ratios of scene text images. Global_CNN_Segment indicates that the segmented image is used as an input; while Global_CNN_Resize indicates that the image is directly resized to a specific size of 40×120 as an input. The results in Table 7 show that our image segmentation method improves the overall accuracy for scene text images. For CVSI-2015, the accuracies of the two methods are the same because only a very small percentage of images are segmented. There are only 57 images that needed to be segmented using the training set, while the total number of images is 6412. This also shows that our segmentation method works well when the aspect ratio greatly changes which is an important feature of natural scene text images. There are only a few instances of scene text images in CVSI-2015 dataset, and so our method is not effective for CVSI-2015 dataset.

4) IMPACT OF DIFFERENT PROCESSING METHODS TOWARDS LOCAL AND GLOBAL FEATURES

We use ResNet-20 as the basic architecture and add a feature-merge branch that is used to combine the local features and global features of a scene text image together. We merge the features of conv_2x and conv_3x using down sampling. In addition, we resize the image to 40×120 to be used as an input. We compare the overall classification accuracy of this method (CNN_feature_merge) with ours. The results

TABLE 8. Comparison of overall accuracy of the method using one CNN to integrate local and global features and ours.

Method	MLe2e	SIW-13	ICDAR-2017	CVSI-2015
CNN_feature_merge	0.871	0.925	0.893	0.955
Ours	**0.958**	**0.961**	**0.932**	**0.983**

in Table 8 show that our method has better performance. There are three advantages of our approach. The first is that our approach using patches and segmented images as inputs avoids problems that are caused by the extremely variable aspect ratios of scene text images. The second is that the design of Local CNN achieves excellent performance, and the last is that we integrate Local CNN and Global CNN to fully exploit the local and global features of images for script identification.

5) FAILURE SAMPLES OF PROPOSED APPROACH

We list some image samples that our approach fails to identify. From Figure 8 we find that our approach fails to identify the scripts in some scene text images with the following conditions. These conditions include obstructed text, complex backgrounds, background and text being mixed together, very poor quality text that is invisible to the naked eye, non-horizontal text, too large of text spacing, and high similarity between the script types. Our follow-up work will focus on these scene text images, especially on identifying the scripts with high similarity in natural scene images.

V. CONCLUSION

In this work, we propose a new framework integrating Local CNN and Global CNN via the Adaboost algorithm for script identification. The architectures of Local CNN and Global CNN are both based on ResNet-20. The experiments show that our approach has outstanding performance on four public datasets. The main reasons are as follows: (1) the advanced ResNet that is used as the basic network architecture of Local CNN and Global CNN, (2) the input strategy and fine-tuning strategy of Local CNN and Global CNN, and (3) the Adaboost algorithm that is used to conduct the decision-level fusion of the results of Local CNN and Global CNN.

REFERENCES

[1] Q. Ye and D. Doermann, "Text detection and recognition in imagery: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 7, pp. 1480–1500, Jul. 2015.

[2] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading text in the wild with convolutional neural networks," *Int. J. Comput. Vis.*, vol. 116, no. 1, pp. 1–20, 2016.

[3] W. Lu, H. Sun, J. Chu, X. Huang, and J. Yu, "A novel approach for video text detection and recognition based on a corner response feature map and transferred deep convolutional neural network," *IEEE Access*, vol. 6, pp. 40198–40211, 2018.

[4] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Deep features for text spotting," presented at the 13th Eur. Conf. Comput. Vis., Zurich, Switzerland, Sep. 2014.

[5] D. Ghosh, T. Dube, and A. Shivaprasad, "Script recognition—A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2142–2161, Dec. 2010.

[6] A. K. Singh and C. V. Jawahar, "Can RNNs reliably separate script and language at word and line level?" presented at the 13th Int. Conf. Document Anal. Recognit., Nancy, France, Aug. 2015.

[7] T. Q. Phan, P. Shivakumara, Z. Ding, S. Lu, and C. L. Tan, "Video script identification based on text lines," presented at the 11th Int. Conf. Document Anal. Recognit., Beijing, China, Sep. 2011.

[8] D. Zhao, P. Shivakumara, S. Lu, and C. L. Tan, "New spatial-gradient-features for video script identification," presented at the 10th IAPR Int. Workshop Document Anal. Syst., Gold Coast, QLD, Australia, Mar. 2012.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," presented at the IEEE Conf. Comput. Vis. Pattern Recognit., Las Vegas, NV, USA, Jun. 2016.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," presented at the Int. Conf. Learn. Represent., San Diego, CA, USA, May 2015.

[11] B. Shi, X. Bai, and C. Yao, "Script identification in the wild via discriminative convolutional neural network," *Pattern Recognit.*, vol. 52, pp. 448–458, Apr. 2016.

[12] L. Gomez, A. Nicolaou, and D. Karatzas, "Improving patch-based scene text script identification with ensembles of conjoined networks," *Pattern Recognit.*, vol. 67, pp. 85–96, Jul. 2017.

[13] A. K. Bhunia, A. Konwer, A. K. Bhunia, A. Bhowmick, P. P. Roy, and U. Pal, "Script identification in natural scene image and video frames using an attention based convolutional-LSTM network," *Pattern Recognit.*, vol. 85, pp. 172–184, Jan. 2019.

[14] F. Huang, X. Li, S. Zhang, J. Zhang, J. Chen, and Z. Zhai, "Overlapping community detection for multimedia social networks," *IEEE Trans. Multimedia*, vol. 19, no. 8, pp. 1881–1893, Aug. 2017.

[15] J. Zhu, H. Zou, S. Rosset, and T. Hastie, "Multi-class AdaBoost," *Statist. Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[16] N. Sharm, R. Mandal, R. Sharma, U. Pal, and M. Blumenstein, "ICDAR2015 competition on video script identification (CVSI 2015)," presented at the 13th Int. Conf. Document Anal. Recognit., Nancy, France, Aug. 2015.

[17] N. Nayef *et al.*, "ICDAR2017 robust reading challenge on multi-lingual scene text detection and script identification-RRC-MLT," presented at the 14th Int. Conf. Document Anal. Recognit., Kyoto, Japan, Nov. 2017.

[18] K. Ubul, G. Tursun, A. Aysa, D. Impedovo, and G. Pirlo, "Script identification of multi-script documents: A survey," *IEEE Access*, vol. 5, pp. 6546–6559, Mar. 2017.

[19] N. Sharma, S. Chanda, U. Pal, and M. Blumenstein, "Word-wise script identification from video frames," presented at the 12th Int. Conf. Document Anal. Recognit., Washington, DC, USA, Aug. 2013.

[20] N. Sharma, U. Pal, and M. Blumenstein, "A study on word-level multi-script identification from video frames," presented at the Int. Joint Conf. Neural Netw., Beijing, China, Jul. 2014.

[21] B. Shi, C. Yao, C. Zhang, X. Guo, F. Huang, and X. Bai, "Automatic script identification in the wild," presented at the 13th Int. Conf. Document Anal. Recognit., Nancy, France, Aug. 2015.

[22] J. Mei, L. Dai, B. Shi, and X. Bai, "Scene text script identification with convolutional recurrent neural networks," in *Proc. IEEE Int. Conf. Pattern Recognit.*, Dec. 2016, pp. 4053–4058.

[23] L. Gómez and D. Karatzas, "A fine-grained approach to scene text script identification," presented at the 12th IAPR Workshop Document Anal. Syst., Santorini, Greece, Apr. 2016.

[24] M. Verma, N. Sood, P. P. Roy, and B. Raman, "Script identification in natural scene images: A dataset and texture-feature based performance evaluation," presented at the Int. Conf. Comput. Vis. Image Process., vol. 460. Singapore, Springer, 2017.

[25] A. Singh, A. Mishra, P. Dabral, and C. Jawahar, "A simple and effective solution for script identification in the wild," presented at the 12th IAPR Workshop Document Anal. Syst., Santorini, Greece, Apr. 2016.

[26] O. K. Fasil, S. Manjunath, and V. N. M. Aradhya, "Word-level script identification from scene images," in *Proc. 5th Int. Conf. Frontiers Intell. Comput., Theory Appl.*, Mar. 2017, pp. 417–426.

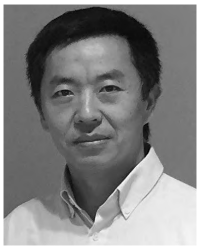
[27] A. Nicolaou, A. D. Bagdanov, L. Gomez-Bigorda, and D. Karatzas, "Visual script and language identification," presented at the 12th IAPR Workshop Document Anal. Syst., Santorini, Greece, Apr. 2016.

[28] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," presented at the 22nd ACM Int. Conf. Multimedia, 2014.

[29] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proc. Int. Conf. Comput. Vis. Pattern Recognit.*, Columbus, OH, USA, Jun. 2014, pp. 1725–1732.

[30] D. G. Lowe, "Object recognition from local scale-invariant features," presented at the 7th Int. Conf. Comput. Vis., Corfu, Greece, Sep. 1999.

LIQIONG LU is currently pursuing the Ph.D. degree with the School of Printing and Packaging, Wuhan University. Her research interests include the text analysis in natural scene images, including text detection, text recognition, and script identification.



YAOHUA YI received the M.S. and Ph.D. degrees from Wuhan University, Wuhan, China, in 2000 and 2004, respectively, where he is currently a Professor with the School of Printing and Packaging. His research interests include image intelligent processing, information extraction technology, imaging quality detection, and analysis technology. He has published over 50 papers in refereed conferences and journals.

FALIANG HUANG received the Ph.D. degree in computer science from the South China University of Technology, Guangzhou, China, in 2011. He is currently an Associate Professor with Fujian Normal University, China. His current research interests include data mining and machine learning.

KAILI WANG is currently pursuing the Ph.D. degree with the School of Printing and Packaging, Wuhan University. Her current research interests include scene text analysis and recognition, image processing, deep learning, and computer vision.

QI WANG received the Ph.D. degree from Nanjing Forestry University, in 2013, where she is currently an Associate Professor with the School of Light Industry and Food Engineering. She has published over 40 papers in refereed conferences and journals. Her research interests include graphic information processing and pre-press quality control technology.