A Sequence Labeling Convolutional Network and Its Application to Handwritten String Recognition*

Qingqing Wang, Yue Lu
Shanghai Key Laboratory of Multidimensional Information Processing
Department of Computer Science and Technology, East China Normal University
3663 North Zhongshan Road, Shanghai 200062, P. R. China
ylu@cs.ecnu.edu.cn

Abstract

Handwritten string recognition has been struggling with connected patterns fiercely. Segmentation-free and over-segmentation frameworks are commonly applied to deal with this issue. For the past years, RNN combining with CTC has occupied the domain of segmentation-free handwritten string recognition, while CNN is just employed as a single character recognizer in the over-segmentation framework. The main challenges for CNN to directly recognize handwritten strings are the appropriate processing of arbitrary input string length, which implies arbitrary input image size, and reasonable design of the output layer. In this paper, we propose a sequence labeling convolutional network for the recognition of handwritten strings, in particular, the connected patterns. We properly design the structure of the network to predict how many characters present in the input images and what exactly they are at every position. Spatial pyramid pooling (SPP) is utilized with a new implementation to handle arbitrary string length. Moreover, we propose a more flexible pooling strategy called FSPP to adapt the network to the straightforward recognition of long strings better. Experiments conducted on handwritten digital strings from two benchmark datasets and our own cell-phone number dataset demonstrate the superiority of the proposed network.

1 Introduction

Handwritten string recognition has attracted intensive attention in recent years due to its vast applications in both industrial projects and financial transactions, such as mail sorting system and bank check processing. A straightforward solution is to segment strings into components which correspond to single characters or part of them, and then analyze the recognition results of each component or their combination to calculate the optimal integrated result. This traditional strategy is called over-segmentation framework. The alternative methods resort to segmentation-free framework, which directly recognizes the whole input strings without segmenting. For example, some researchers try to obviate segmenting by utilizing Recurrent Neural Network (RNN)/Long Short-Term Memory (LSTM) combining with Connectionist Temporal Classification (CTC) in the task of handwritten string recognition. This strategy has been flourishing since the revival of deep learning due to its integrated structure. For these two frameworks, the former requires sophisticated techniques about over segmenting, touch splitting, single character recognizing and best path searching, while the latter needs plenty of training data.

Liu's group has proposed many efficient algorithms based on the over-segmentation framework such as [Liu *et al.*, 2004] [Wang *et al.*, 2012]. Particularly, one of their algorithms won the first place on the ICFHR2014 handwritten digit string recognition (HDSR) competition [Diem *et al.*, 2014]. Keysers et al. [Keysers *et al.*, 2016] presented Google's online handwriting recognition system that currently supports 22 scripts and 97 languages. This system also followed the over-segmentation framework and has been publicly available in several Google products such as Google Translate. In these methods, classifiers are trained only for single characters, therefore at the previous stage, it is expected to form intact single characters by over segmenting strokes and combing consecutive ones. Segmentation-free methods based on RNN/LSTM/CTC also achieved comparable performance in the ICFHR2014 HDSR competition [Diem *et al.*, 2014] and showed its superiority to the winner on the most challenging CAR-B dataset. Additionally, this kind of methods are also widely used in other languages such as English [Graves *et al.*, 2009], Arabic [Graves and Schmidhuber, 2008], Chinese [Messina and Louradour, 2015] and so forth. It is remarkable that, on the recent ICFHR2016 handwritten text recognition competition [Sanchez *et al.*, 2016], all of the six submissions are RNN/LSTM/CTC based. In these methods, RNN maps current input to corresponding output with considering its history (namely previous inputs), and CTC is assembled at the output layer to perform sequence labeling without requirement of pre-segmenting inputs and post-processing network's outputs. To cope with the problem of gradient vanishing, LSTM is proposed as an improvement of

*Corresponding author (ylu@cs.ecnu.edu.cn). This work is jointly supported by the Science and Technology Commission of Shanghai Municipality under research grant 14511105500 and the Natural Science Foundation of Shanghai No. 17ZR1408200.

the original RNN.

Convolutional Neural Network (CNN) is a popular topic in the field of deep learning and has played dominant roles in many tasks. Numerous sophisticated techniques are studied to promote its expression ability and recognition accuracy. However, few strategies are proposed for sequence labeling with CNN except Goodfellow et al. [Goodfellow *et al.*, 2014]. The main difficulties are posed by the arbitrary input size and the reasonable structure design. Goodfellow et al. [Goodfellow *et al.*, 2014] recognized Street View House Number (SVHN) with CNN by assembling multiple softmax classifiers at the output layer. The first one was employed to predict string length and its prediction result determined how many following classifiers should be taken to figure out target string. But this method separately trained individual classifiers, and saved separate weight matrix for each separate digit classifier. As discussed in [Goodfellow *et al.*, 2014], for long sequences this could incur too high of a memory cost. Moreover, this method resized input images into a fixed size and alleviated deformation caused by resizing with additional background pixels. This operation also limited the capability of this method. As pointed out by Goodfellow et al., for large maximum length, their method was unlikely to scale well. Besides, the resizing operation is not suitable for handwritten string recognition since the lengths of handwritten strings are significant different, and any resizing operation can cause serious deformation of characters. And in this case, easing deformation by padding additional background pixels will result in heavy computation burden. SPP proposed by He et al. [He *et al.*, 2015] has the capability to handle the arbitrary input sizes without resizing operation. Unfortunately, in order to take advantage of existing GPU implementations (such as *cuda-covnet* and *caffe*), He et al. implemented their SPP network by two fixed-size networks sharing parameters and preserved the SPP behaviors by multiple conventional sliding window pooling layers with different pooling sizes and strides. Hence, during the training procedure, training samples still needed to be resized into fixed sizes.

In this paper, we carefully design a sequence labeling convolutional network to recognize handwritten strings, in particular, the intractable connected patterns. SPP is employed with a new implementation so that training samples even in the same batch can own different sizes. For the original SPP, images with different aspect ratios, which indicate different string lengths, share the same scale setting. Intuitively, more features should be extracted for images containing more characters, while fewer features are enough for those with fewer characters. Toward this end, we propose a more flexible pooling strategy called FSPP, which fixes the vertical scale setting and designs different horizon scale setting to images according to their aspect ratios.

To demonstrate the effectiveness of the proposed network, we conduct experiments on handwritten digit strings from CVL and ORAND-CAR datasets. Comparison with all the participating methods of ICFHR2014 competition shows the superiority of our proposed network. Furthermore, we collect a cell-phone number dataset named PhPAIS(cell-phone number dataset collected by Lab of Pattern Analysis and Intelligence System) from the real China post images to verify the practicability of FSPP.

2 Proposed Network

Architecture of the proposed network is depicted in Figure 1. Four convolutional layers and two mean pooling layers are designed before an Inception module, which is followed by a SPP or FSPP layer. Then the fixed-length features extracted by the SPP or FSPP layer are fed into two fully-connected layers that assembled before the final classifier layer. If we regard SPP or FSPP layer as extracting global features at multiple scales, and Inception module as extracting local features at multiple scales, setting a SPP or FSPP layer after an Inception module [Szegedy *et al.*, 2015] will be beneficial for the model to extract richer information. The goal of the target network is to predict the order concerned sequential characters. Intuitionally, information from two aspects needs to be extracted: (1) how many characters are contained in the input images, and (2) what exactly it is at each position. So we equip the proposed network with two modules named CM, which is short for 'counting module', and PM, which is short for 'prediction module', to predict corresponding information. Apparently, the proposed network is supposed to have a cluster of classifiers rather than a single one. Therefore, we need to blend prediction errors of these classifiers into one structure when we design the cost function for the network.

2.1 Model Prediction

Considering an input image X containing N characters $D = \{d_1, d_2, ..., d_N\}$, the objective of the basic approach is to predict N and digit d_i at position p_i correctly. Suppose the maximal input sequence length as K, which is pre-determined, we design K nodes at CM and K softmax classifiers at PM. Let's denote the prediction result of CM as L, $L \leqslant K$, and K prediction results of PM as $S = \{s_1, ..., s_K\}, s_i \in \Phi \cup \{NAN\}$, where Φ represents the alphabet of characters needed to be predicted and *NAN* means there is no character showing at this position. In this architecture, we assume that characters are independent from each other. So for a specific sequence with l characters $G = \{l, g_1, ..., g_l\}, l \leq K$, its probability output can be defined as

$$P(G|X) = p(L = l|X) \prod_{i=1}^{l} p(s_i = g_i|X) \prod_{j=l+1}^{K} p(s_j = NAN|X) \tag{1}$$

where $p(\cdot)$ is the probability output of softmax classifiers. The optimal prediction result $G' = \{l', g'_1, ..., g'_{l'}\}$ of the architecture is determined by

$$G' = (l', g'_1, ..., g'_{l'}) = \underset{l, g_1, ..., g_l}{\operatorname{argmax}} \, logP(G|X) \tag{2}$$

At backward propagating stage, the gradients are back propagated with Stochastic Gradient Descent (SGD) with momentum. We denote the batch size as m, then the cost function of the proposed network can be formulated as

$$J = -\frac{1}{m}[\sum_{i=1}^{m} log(p(L = N^{(i)}|X^{(i)}) \prod_{k=1}^{N^{(i)}} p(s_k^{(i)} = d_k^{(i)}|X^{(i)}) \prod_{k=N^{(i)}+1}^{K} p(s_k^{(i)} = NAN|X^{(i)}))] \tag{3}$$

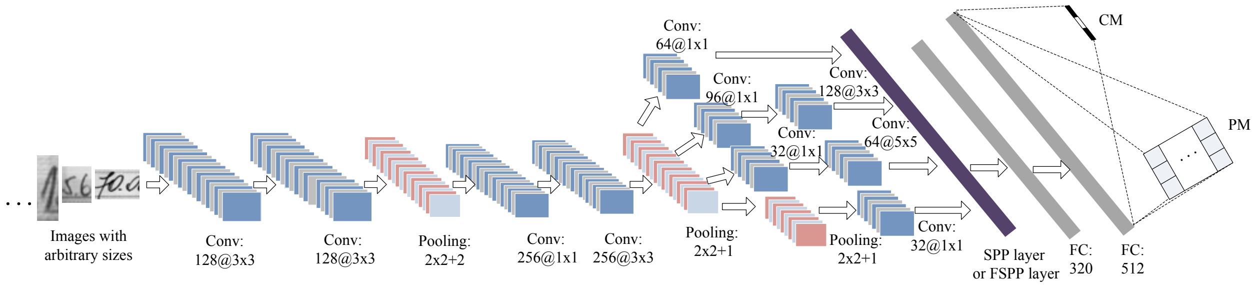


Figure 1: Structure of the proposed network

where $X^{(i)}$ is the input image of the ith sample, which contains $N^{(i)}$ characters $\{d_1^{(i)}, ..., d_{N^{(i)}}^{(i)}\}$, and $s_k^{(i)}$ denotes the prediction result of the kth classifier in PM for the ith sample. This cost function fuses prediction errors of CM and PM into one structure, so we can train all of the softmax classifiers simultaneously by minimizing this function. In order to prevent overfitting, we introduce a weight decay term and rewrite the cost function as

$$
\begin{aligned}
J = -\frac{1}{m}[\sum_{i=1}^{m} log(p(L = N^{(i)}|X^{(i)}) \prod_{k=1}^{N^{(i)}} p(s_k^{(i)} = d_k^{(i)}|X^{(i)}) \\
\prod_{k=N^{(i)}+1}^{K} p(s_k^{(i)} = NAN|X^{(i)}))] + \frac{\lambda}{2}\sum w^2 \\
s.t. \quad w \in W
\end{aligned}
\tag{4}
$$

where λ is the weight decay used to control the relative importance of this item, and W denotes the set of weight coefficients of the network. Parameters θ including weight coefficients W and bias b are updated by $\theta = \theta - \alpha\nabla_\theta J$, where α is the learning rate.

In our design, possibilities of strings with length from 1 to K are calculated and compared with considering all of the classifiers' prediction results. While in Goodfellow's design, if the prediction result of the first classifier is k, the prediction results of the k following classifiers are simply concatenated to obtain the target string and prediction results of other classifiers are omitted. Additionally, unlike Goodfellow's work, which resizes input images into a fixed size, We utilize the re-implemented SPP or FSPP to handle arbitrary input image sizes without any resizing operation.

2.2 Implementation of SPP

The input of the SPP layer is a serial of feature maps, each of which needs to be pooled into fixed-length representations at this layer in the way depicted in Figure 2(a). An input feature map is divided into different bins evenly at each scale, and for the tth scale, there will be $t \times t = t^2$ bins. Then for each bin, we calculate the mean value or max value (mean value is chosen in our experiments) as its representation. Obviously, if we set the overall scale number as T, $1 + 2^2 + ... + T^2 = \frac{T(T+1)(2T+1)}{6}$ features will be obtained for each feature map. We denote the size of an input feature map as $a \times b$. For the tth scale, if a or b cannot be exactly divided by t, bins of the last column or row will share pixels with the previous ones.

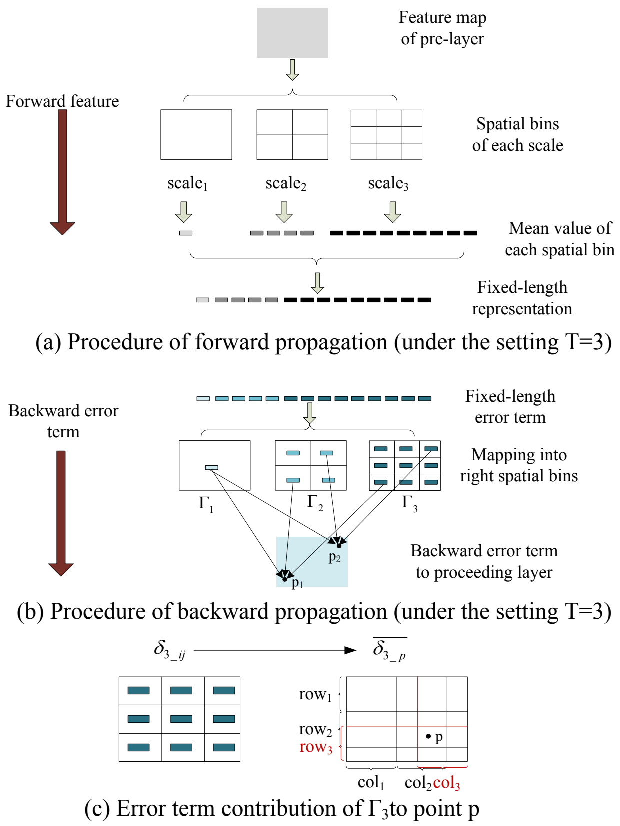

Feature map of pre-layer
Forward feature
Spatial bins of each scale
scale$_1$
scale$_2$
scale$_3$
Mean value of each spatial bin
Fixed-length representation
(a) Procedure of forward propagation (under the setting T=3)
Backward error term
Fixed-length error term
Mapping into right spatial bins
Γ_1
Γ_2
Γ_3
p$_1$
p$_2$
Backward error term to proceeding layer
(b) Procedure of backward propagation (under the setting T=3)
δ_{3_ij}
$\overline{\delta_{3\_p}}$
row$_1$
row$_2$
row$_3$
p
col$_1$
col$_2$
col$_3$
(c) Error term contribution of $\Gamma_3$to point p

Figure 2: Implementation of spatial pyramid pooling

The procedure of back propagating error terms is described in Figure 2(b). Firstly, elements of error term vector are mapped into corresponding spatial bins of each scale. Then error terms of proceeding layer are calculated from values of bins at different scales. For the tth scale, we denote error term of bin at position (i, j) as δ_{t_ij}, and the error term map of this scale as Γ_t. $\overline{\delta_{t\_p}}$ means the total error term contribution of bins at tth scale to pixel p of certain feature map in previous layer. The final error term of pixel p is calculated by

$$
\delta p = \sum_{i=1}^{T} \frac{\overline{\delta_{i\_p}}}{\lfloor a/i \rfloor \times \lfloor b/i \rfloor} \tag{5}
$$

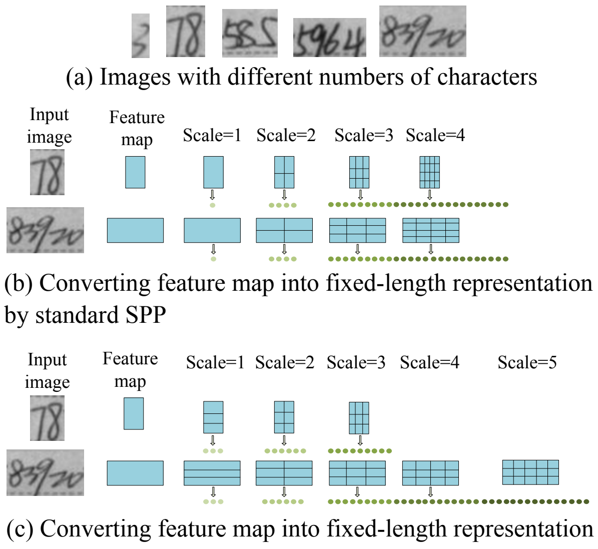
Note that in the forward procedure, several bins may share same pixels. So error terms are also shared by multiple pixels in the corresponding backward procedure. Figure 2(c) demonstrates one of the sharing cases. In this situation, $\overline{\delta_{3\_p}}$ is computed by (6) since there are four bins sharing pixel p in the forward procedure.

$$\overline{\delta_{3\_p}} = \delta_{3\_22} + \delta_{3\_33} + \delta_{3\_32} + \delta_{3\_23} \quad (6)$$

With above implementation, our network can be trained with samples in arbitrary sizes directly. In the experiments, samples within the same batch may have different sizes, which is different from the implementation in [He *et al.*, 2015]. Theoretically, our implementation can be used to recognize handwritten strings with any length, while the original training strategy proposed in [He *et al.*, 2015] cannot since its resizing step will cause deformation of strings. And the longer the string is, the more serious the deformation becomes.

2.3 FSPP Layer

FSPP is different from SPP in the following two aspects: (1) in the vertical direction, the scale is fixed (set to 3 in our work), and (2) in the horizontal direction, the setting of scale is dynamically changed according to the aspect ratios of input images. The procedure of feature extraction is shown in Figure 3. As we can see, in the SPP strategy, 30 features are extracted for both images containing '78' and '83920' when the sale is set to 4. While in the FSPP strategy, 18 features are extracted for the former and 45 features are extracted for the latter when the scale is set to 3 and 5 for them, respectively.



(a) Images with different numbers of characters

(b) Converting feature map into fixed-length representation by standard SPP

(c) Converting feature map into fixed-length representation by the proposed FSPP

Figure 3: Feature extraction of the SPP layer and FSPP layer from feature maps with arbitrary sizes

Then we send the extracted features into the next fully-connected layer by the way shown in Figure 4, where FM_i represents the converted features of ith feature map. In this figure, the scale is set to 2, 3 and 4 (or larger) for images with aspect ratio (denoted by AR) within the interval $(0, \tau_1]$,

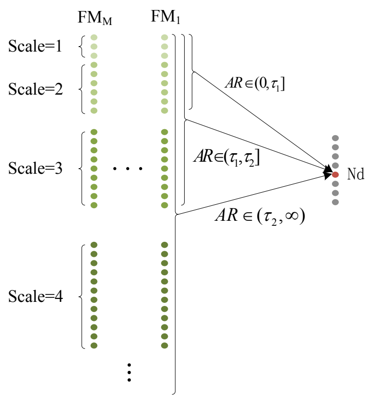


Figure 4: Connection of the FSPP layer to the next fully-connected layer

$(\tau_1, \tau_2]$ and (τ_2, ∞), respectively. Each node Nd in the next layer needs to connect with all of the extracted features in the FSPP layer. For example, if the aspect ratio of certain input image is within $(\tau_1, \tau_2]$, the scale will be set as 3 in the FSPP layer, and each node Nd will connect with $3 \times (1 + 2 + 3) \times M = 18M$ features totally. In practice, we need the image sizes together with raw image pixels flow into the network together so that the GPU can determine pooling scales for individual images at this layer directly.

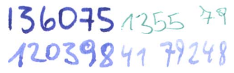
3 Experiments

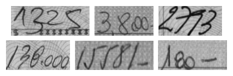
We conduct experiments on handwritten digit strings obtained from CVL and ORAND-CAR datasets (provided by Diem et al. [Diem *et al.*, 2014]) to demonstrate the effectiveness of the proposed network, and compare its performance with all the participating methods of ICFHR2014 HDSR competition. To further verify the practicability of the proposed network and the priority of the FSPP layer, we carry out experiments on our own cell-phone number dataset named PhPAIS. Figure 5 exhibits some samples, where ORAND-CAR is split into CAR-A and CAR-B since its images are obtained from real checks provided by two banks, and some digits of the PhPAIS dataset are covered for privacy protection. Distribution of the samples in each dataset with respect to string length is shown in Table 1.

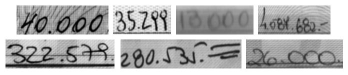
Our network uses Rectified Linear Units (ReLU) as activation function, and the learning rate initialized at 0.01 drops 50% after each epoch. The precision is defined as the number of correctly recognized strings divided by the total number of strings, which is the same as the hard metric defined in ICFHR2014 competition [Diem *et al.*, 2014]. The system is coded in Matlab, C++ and CUDA language without using any other open framework like *caffe*.

3.1 Training of the Proposed Network

From Table 1 we can see that the length of digit strings ranges from 2 to 11. But it does not mean all of the digits are totally connected (as shown in Figure 5). Actually, handwritten


(a) CVL


(b) CAR-A


(c) CAR-B

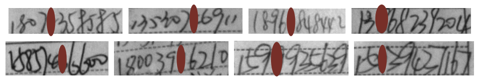

(d) PhPAIS

Figure 5: Samples from each dataset

Table 1: Distribution of the four different databases with respect to string length

	train				test			
Len	CVL	CAR-A	CAR-B	PhP-AIS	CVL	CAR-A	CAR-B	PhP-AIS
2	0	22	0	0	0	36	0	0
3	0	204	0	0	0	387	5	0
4	0	704	63	0	0	1425	69	0
5	125	903	1200	0	789	1475	1241	0
6	758	145	1599	828	4144	363	1452	912
7	379	29	137	244	1765	87	157	350
8	0	2	1	264	0	11	2	379
9	0	0	0	178	0	0	0	313
10	0	0	0	364	0	0	0	586
11	0	0	0	3122	0	0	0	3937
total	1262	2009	3000	5000	6698	3784	2926	6477

digit strings usually consist of single digits and connect patterns with shorter length, and it is not hard to split these digit strings. Therefore, our network only needs to deal with the single digits and the connected patterns. According to our statistics, the dominant lengths of connected patterns in CVL and CAR datasets is 1, 2 and 3, so it is reasonable to set the maximal input sequence length $K = 3$ for CVL and CAR datasets. The connection situation of PhPAIS is more serious, therefore we set $K = 5$ for this dataset.

To train the proposed network, we generate new datasets by manually cropping the digit images from CVL, CAR-A, CAR-B and PhPAIS, and naming them as croppedCVL, croppedCAR-A, croppedCAR-B and croppedPhPAIS, respectively. The new datasets are composed of single digits and strings with 1 to 5 digits. Corresponding distribution is presented in Table 2.

Table 3: Recognition accuracies (%) on the cropped datasets

Len	1	2	3	Mean
croppedCVL	96.37	89.21	78.32	89.11
croppedCAR-A	98.06	95.29	89.61	95.09
croppedCAR-B	98.72	95.97	92.65	96.23

Recognition accuracies of the proposed network with respect to different string length on croppedCVL, croppedCAR-A and croppedCAR-B datasets are shown in Table 3. Since the connection situation of these datasets are not very serious, we just use SPP layer in this network and set the scale to 4 in this layer. As we can see, though the background of ORAND-CAR is very complicated, our proposed network is still capable of achieving the accuracies higher than 95% on both croppedCAR-A and croppedCAR-B datasets. We attribute the success to the deep and properly designed network structure and to the huge amount of training samples. Additionally, though CAR-B is the most challenging dataset among CVL, CAR-A and CAR-B, which is summarized by [Diem *et al.*, 2014], our network achieves the best performance on croppedCAR-B dataset in Table 3, especially for strings with length of three. The reason is implied in Table 3, where the amount of training samples in croppedCAR-B is about two times of that in croppedCVL or croppedCAR-A. So we can figure out that more samples are very helpful for improving the performance of the proposed network, and it is very likely to improve the accuracies on croppedCVL and croppedCAR-A by providing more training samples.

3.2 Performance of the Proposed Network

We compare our method with the submissions of ICFHR2014 HDSR competition under the same evaluation metrics. To recognize handwritten strings in CVL and ORAND-CAR, we coarsely segment original images into single digits and connected patterns by using horizontal projection and connected component analysis. Then the trained sequence labeling network is used to recognize each segment and the final results are obtained by simply joining the recognition results of all segments together.

Comparison results are shown in Table 4. Obviously, the proposed method achieves the highest mean accuracy on the

Table 2: Distribution of the cropped datasets with respect to string length

	croppedCVL		croppedCAR-A		croppedCAR-B		croppedPhPAIS	
Len	train	test	train	test	train	test	train	test
1	7613	4714	8985	3667	15515	4386	48248	7133
2	6434	3977	6957	2886	13198	3599	42934	6368
3	5267	3206	4985	2088	10489	2802	37710	5630
4	0	0	0	0	0	0	33318	5211
5	0	0	0	0	0	0	28207	4425
total	19314	11897	20927	8641	39202	10787	190417	28767

Table 4: Recognition accuracies (%) of different methods

Method	CAR-A	CAR-B	CVL	Mean	Framework
Tebessa I	37.05	26.62	59.30	40.99	over-segmentation
Tebessa II	39.72	27.72	61.23	42.89	over-segmentation
Singapore	52.30	59.60	50.40	54.10	segmentation free
Pernamb -uco	78.30	75.43	58.60	70.78	over-segmentation segmentation free
Beijing	80.73	70.13	**85.29**	78.72	over-segmentation
Proposed	**82.61**	**83.32**	79.23	**81.72**	—-

three datasets. Especially, the accuracies on both challenging CAR-A and CAR-B datasets outperform the other methods by a large margin. According to [Diem *et al.*, 2014], Beijing along with Pernambuco outperformed the other participating methods. Mean accuracy on CAR-A and CAR-B datasets of the Beijing and Pernambuco are $(80.73\% + 70.13\%)/2 = 75.43\%$ and $(78.30\%+75.43\%)/2 = 76.87\%$, respectively. While our method gets a mean accuracy of $(82.61\%+83.32\%)/2 = 82.97\%$, which is 7.54% and 6.10% higher than the Beijing and Pernambuco methods. Beijing is a well-designed over-segmentation method and Pernambuco is a combination of multiple classifiers including a hybrid classifier combining k-NN and SVM, and a hybrid classifier combining SVM and MDRNN [Graves and Schmidhuber, 2008]. The combination of k-NN and SVM is supposed to be an over-segmentation method while the combination of SVM and MDRNN is a segmentation-free method. Besides that, Pernambuco takes real data extracted from Brazilian bank checks of last decades into account. Singapore combines HOG feature and RNN to handle this task, but due to the small amount of training samples, it doesn't perform well.

From Figure 5 we can see that both CAR-A and CAR-B are disturbed by background clutter and suffer from character connection problem. It seems there is no difference between these two datasets. However, according to Table 3, the dominant length of CAR-A is four and five, while it is five and six for CAR-B. The little difference has caused large margin in recognition accuracy. As shown in Table 4, all the methods achieved higher accuracy in CAR-A than CAR-B except Singapore. Apparently, these methods are very sensitive to string length. However, our method achieves very similar accuracy on the two datasets, which implies that the proposed method is more robust to string length. We blame the low accuracy of our method on CVL to the lack of diversity of this dataset since only 26 different strings are included by its 7960 samples (only 1262 for training). Though it is 6% lower than Beijing on CVL dataset, our method still achieves an accuracy 20% higher than Pernanmbuco. Overall, our method outperforms all of the participating methods of ICFHR2014 competition with a mean accuracy of 81.72% on the three datasets.

Table 5: Recognition accuracies (%) on the croppedPhPAIS and PhPAIS datasets

		croppedPhPAIS					PhPAIS
Setting	Pooling &scale	Len=1	Len=2	Len=3	Len=4	Len=5	
1	SPP&4	99.45	98.79	97.10	95.30	91.82	84.47
2	SPP&3	99.36	98.38	97.26	94.53	89.99	82.49
3	FSPP &2,3,4	99.44	98.45	97.19	94.59	91.77	84.31
4	FSPP &3,4,5	99.48	98.74	97.39	95.16	92.45	84.72

3.3 Comparison of SPP and FSPP on PhPAIS Dataset

The superiority of the proposed FSPP is evaluated on the PhPAIS dataset. Images in this dataset suffer from much longer string length (mostly 11) and more serious connection situation (maximal string length of connected patterns is up to 5). The proposed network is evaluated under four settings, which share the same configuration except the SPP/FSPP layer. Details and recognition results are shown in Table 5. Aspect ratio intervals of FSPP in setting 3 and 4 are $(0, 0.5)$, $[0.5, 0.9)$ and $[0.9, \infty)$. Network trained on the croppedPhPAIS is used to recognize the full-length strings of the PhPAIS dataset by cooperating with some simple coarse segmentation strategies.

Comparing recognition results of the first two settings we can figure out that, reducing scales of the SPP layer will not cause significant performance degradation to the short strings, which means too many features is redundant and unnecessary for short strings. In contrast, a larger performance degradation can be observed on longer strings. Intuitively, longer strings need richer features to discriminate characters at each position, so increasing scales is preferable to reducing them. Overall, scale reduction leads to a performance degradation of 2% on the PhPAIS dataset. When the more flexible FSPP is employed, different scales are set according to aspect ratios that related to string lengths. Obviously, the performance degradation is alleviated in a large degree and only 0.16% accuracy reduction is caused. Furthermore, we can reducing scales for short strings and increasing scales for long strings at the same time by using FSPP as the forth setting does. By this means, performance is slightly improved when comparing the first and the last setting. In conclusion, the proposed FSPP is a better choice for strings with longer string lengths and more serious connection situation.

4 Conclusions

This paper presents a sequence labeling convolutional neural network for handwritten string recognition. We re-implement the SPP layer to handle the arbitrary string length problem, and design an output layer with multiple softmax classifiers to deal with sequence labeling issue. Additionally, we propose a more flexible pooling strategy called FSPP on the basis of SPP to promote the performance of the proposed network on long strings. The proposed network directly recognize handwritten strings without segmenting them, so the challenges posed by connected patterns are avoided. Experimental results show that by combining with some simple coarse segmentation strategies, the proposed network can be utilized to process longer strings. Theoretically, the proposed network can be generalized to any languages or scripts.

References

[Diem *et al.*, 2014] Markus Diem, Stefan Fiel, Florian Kleber, Robert Sablatnig, Jose M. Saavedra, David Contreras, Juan Manuel Barrios, and Luiz S. Oliveira. ICFHR 2014 competition on handwritten digit string recognition in challenging datasets (HDSRC 2014). *International Conference on Frontiers in handwriting recognition*, pages 779–784, 2014.

[Goodfellow *et al.*, 2014] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv:1312.6082v4, Computer Science*, 2014.

[Graves and Schmidhuber, 2008] Alex Graves and Jurgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. *Advances in Neural Information Processing Systems*, 21:545–552, 2008.

[Graves *et al.*, 2009] Alex Graves, Marcus Liwicki, Santiago Femandez, Roman Bertolami, Horst Bunke, and Jurgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transaction on Pattern Recognition and Machine Intelligence*, 31(5):855–868, 2009.

[He *et al.*, 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transaction on Pattern Recognition and Machine Intelligence*, 37(9):1904–1916, 2015.

[Keysers *et al.*, 2016] Daniel Keysers, Thomas Deselaers, Henry A. Rowley, Li-Lun Wang, and Victor Carbune. Multi-language online handwriting recognition. *IEEE Transaction on Pattern Recognition and Machine Intelligence, DOI: 10.1109/TPAMI.2016.2572693*, 2016.

[Liu *et al.*, 2004] Cheng-Lin Liu, Hiroshi Sako, and Hiromichi Fujisawa. Effects of classifier structures and training regimes on integrated segmentation and recognition of handwritten numeral strings. *IEEE Transaction on Pattern Recognition and Machine Intelligence*, 26(11):1395–1407, 2004.

[Messina and Louradour, 2015] Ronaldo Messina and Jerome Louradour. Segmentation-free handwritten Chinese text recognition with LSTM-RNN. *International Conference on Document Analysis and Recognition*, pages 171–175, 2015.

[Sanchez *et al.*, 2016] Joan Andreu Sanchez, Veronica Romero, Alejandro H. Toselli, and Enrique Vidal. ICFHR2016 competition on handwritten text recognition on the READ dataset. *15th International Conference on Frontiers in Handwriting Recognition*, pages 630–635, 2016.

[Szegedy *et al.*, 2015] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with conclutions. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[Wang *et al.*, 2012] Qiu-Feng Wang, Fei Yin, and Cheng-Lin Liu. Handwritten Chinese text recognition by integrating multiple contexts. *IEEE Transaction on Pattern Recognition and Machine Intelligence*, 34(8):1469–1481, 2012.