Handwritten Digit String Recognition for Indian Scripts

Hongjian Zhan[1(✉)], Pinaki Nath Chowdhury[2], Umapada Pal[2], and Yue Lu[1]

[1] Shanghai Key Laboratory of Multidimensional Information Processing, Department of Computer Science and Technology, East China Normal University, Shanghai 200241, China
ecnuhjzhan@gmail.com
[2] CVPR Unit, Indian Statistical Institute, Kolkata, India

Abstract. In many documents digits/numerals may touch each other and hence digit string recognition is necessary as segmentation of individual numeral from the touching string is difficult. In this paper, we propose a digit string recognition system for four Indian popular scripts. Here we consider strings of Kannada, Oriya, Tamil and Telugu scripts for our experiment. This paper has two contributions: (i) we have developed 4 datasets of digit string for each of these four scripts. Each dataset has 20000 numeral string samples for training and 30000 samples for testing. As there is no such dataset available, it will be helpful to the community (ii) we apply a RNN free CNN (Convolutional Neural Network) and CTC (Connectionist Temporal Classifica-tion) based architecture for numeral string recognition. Unlike normal text string, in string of digits has no contextual information among the digits and hence a digit may be followed by an arbitrary digit in a digit string. Because of such behaviors we apply a CNN and CTC based architecture without RNN for numeral string recognition. We tested our scheme on our different test datasets and results are provided.

Keywords: String recognition · Convolutional Neural Network · Connectionist Temporal Classification · Postal Automation

1 Introduction

Because of various applications, recognition of handwritten numeral string has been a popular area for many years to the researchers. Some of its potential application areas are Postal Automation, Bank cheque processing, etc. Although India is a multi-lingual and multi-script country (in India there are about 25 official languages and 11 different scripts are used to write these languages) but not much work is done towards the string recognition of Indian handwritten numerals [2,18,19].

There are two approaches for handwriting numeral string recognition. One is segmentation based and other is segmentation-free [1]. In many Indian documents, digits may touch in different manners like top touching, middle-touching

S. Palaiahnakote et al. (Eds.): ACPR 2019, LNCS 12047, pp. 262–273, 2020.
https://doi.org/10.1007/978-3-030-41299-9_21

and bottom touching. Such touching may be categorized as single point touching, multiple point touching, ligature point touching etc. Moreover, two, three, four and five digit touching strings are also available in Indian documents and hence it is very difficult for accurate segmentation of individual digits from such touching string.

Although there are many pieces of work on Indian isolated digit recognition, there is not much work on Indian digit string recognition. Related pieces of Indian isolated digit recognition work can be seen in [2–9]. As this work is on string recognition, we briefly described here the existing work on digit string recognition.

To recognize Indian pin code written in Pal et al. [18] proposed a segmentation free segmentation approach. Here, at first, binarization of the input document is done. Next, water reservoir concept is applied to pre-segment a pin code string into possible primitive components (individual digits or its parts). Pre-segmented components [10] of the pin code are then merged into possible digits to get the best pin code using dynamic programming (DP) and modified quadrat-ic discriminant function (MQDF) [11] classifier. In 2009, Pal et al. [19] also pro-posed a technique for Bangla, Hindi and English digit-string recognition system.

Although there are several pieces of work on isolated numeral recognition for Kannada, Oriya, Tamil and Telugu scripts [2], to the best of our knowledge there is no work on numeral string recognition for any of these four scripts. Hence in this paper we have considered digit string recognition of these four scripts for our experiments. Also no datasets are available for digit string for these four scripts and hence we have proposed several datasets for this work.

Unlike normal text string, in string of digits there is no contextual information among the digits as a digit may be followed by an arbitrary digit in a string of digits. Because of this, in this paper, we propose a new architecture which is based on CNN (Convolutional Neural Network) and CTC (Connectionist Temporal Classification) [13], without using RNN for numeral string recognition. Also to connect CNN with CTC, we transform the outputs of CNN to a two-dimension vector to meet the feeding requirement of CTC. Furthermore, we utilize dense blocks to build CNN part to extract efficient image features.

Rest of the paper is organized as follows. In Sect. 2 we discuss about the properties of Indian scripts considered here. Section 3 deals with dataset details. Proposed methodology is presented in Sect. 4. The experimental results are discussed in Sect. 5. Finally, conclusion is given in Sect. 6.

2 Properties Indian Scripts

Most of the Indian scripts are originated from Brahmi script through various transformations. Writing style of the Indian scripts considered in this paper is from left to right, and concept of upper/lower case is absent in these scripts.

Oriya is a popular language and script of India. This language is used mainly in the Odisha (formerly Orissa) state of India and also in West Bengal, Jharkhand, and Gujarat. Oriya is the official language of Odisha state.

Kannada is another popular script and it is the official language of the southern Indian state, Karnataka. Kannada is a Dravidian language mainly used by the people of Karnataka, Andhra Pradesh, Tamil Nadu and Maharashtra.

Telugu is the 3rd most popular scripts in India. It is the official language of the southern Indian state, Andhra Pradesh. Telugu is also spoken in Bahrain, Fiji, Malaysia, Mauritius, Singapore and the UAE. The Telugu script is closely related to the Kannada script.

Tamil is also a popular south Indian Language and one of the oldest languages in the world. It is the official language of the southern Indian state, Tamil Nadu. Apart from India, it is also one of the official languages in the countries like Singapore, Malaysia and Sri Lanka.

To get an idea about digit shapes of the four scripts considered in this paper, a set of handwritten samples of these scripts is shown in Fig. 1.

Fig. 1. Handwritten numeral samples of four scripts.

The challenging part of Indian script handwritten recognition is the distinction between the similar shaped components. Sometimes a very small part is the distinguishing mark between two numerals. These small distinguishing parts increase the recognition complexity and decrease recognition accuracy. Because of the writing styles of different individuals, same numerals may take different shapes and conversely two or more different numerals of a script may take similar shape. These factors also increase the complexity of recognition method. To get the idea of similar shape numerals, we provide some examples in Fig. 2. Here in the first row both the numerals of the first pair of Telugu script look like zero. But the first numeral of this pair is Telugu 'one' and 2nd numeral is Telugu 'zero'. Similarly, in the second pair, it seems both the digits are similar. But first digit is '6' and second digit is '9'.

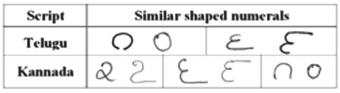


Fig. 2. Examples of some similar shaped numerals.

3 Data Collection

Deep Learning models requires large amount of data for training. Moreover, in many of the Indian scripts no large numeral database is available for deep learning purpose. As there are 11 different scripts are available for India, it is difficult to develop large datasets for each of these 11 scripts. Although some large handwritten numeral string datasets are available for Bangla and Hindi, no large handwritten numeral/digit string dataset is available for Kannada, Oriya, Tamil and Telugu. Hence, here we have developed 4 datasets of handwritten digit string for each of these four scripts. Each dataset has 20000 handwritten digit string samples for training and 30000 samples for testing. The dataset contains numeral string samples of length 6 digits to 10 digits uniformly distributed throughout the training and testing set.

To make the dataset of various complexities, we generate 4 datasets for each script and they are named as Dataset-1, Dataset-2, Dataset-3, and Dataset-4. The first dataset, i.e. "Dataset-1" has only non-touching or non-overlapping numeral whereas Dataset-2 consists of numeral string which may or may not be touching/overlapping. Both real digit string and synthetic digit string are there in these two datasets. By synthetic digit string we mean the digit strings that are generated through computer program from real isolated handwritten digits.

Other two datasets (i.e. "Dataset-3" and "Dataset-4") are completely synthetic and all the digits in a string are touching/overlapping. These datasets are generated through computer program from real isolated handwritten digits and the touching/overlapping area is different in these two datasets. "Dataset-3" has overlapping digit string where there is a maximum overlap of 1 stroke width and "Dataset-4" consists of digit strings having a maximum overlap of 2 stroke width. Here stroke width of the digit which is going to be included in the string during digit string formation is considered for touching. Examples of each of the datasets of each script are shown in Fig. 3(a–d).

During synthetic digit string generation, each digit strings were randomly generated ensuring that the individual digits in a particular sample have similar size and stroke width.

As mentioned earlier, we have four datasets in each scripts and hence there are total 16 datasets. As mentioned above, each dataset has 20000 handwritten digit string samples for training and 30000 samples for testing. That means we have a total of 50,000 samples, including training and testing set, for each dataset of a script. Thus, for 16 datasets we have a total of 800,000-digit string samples.

Now these datasets are available freely to the researchers. As there so such big datasets, we hope this dataset will be helpful to the researchers and the generated dataset can be used as a test bed for performance evaluation of on digit string recognition.

4 Methodology

We propose a new network for handwritten digit string recognition, which is shown in Fig. 4. In order to enhance the performance of DenseNet [12], we add residual connections between dense blocks, For the output layer, we apply a CTC [13] to calculate loss at the training phase and give the predictions at the testing phase.

4.1 Dense Block

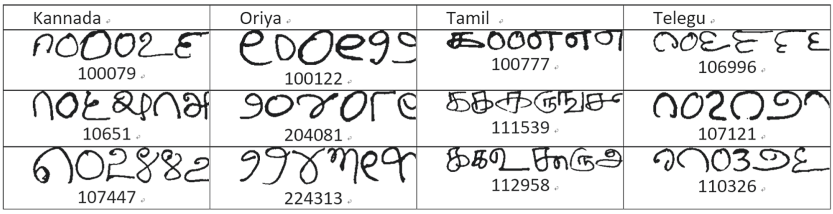
A dense block is a stack of densely connected convolutional layers [12]. It can extract more efficient features than plain and residual convolutional networks. It consists of a group of layers, the batch normalization layer [15], ReLU layer [16] and a convolutional layer. The kernel size of the convolutional layer is 3*3, which can maintain the size of feature map in the whole dense block. After the convolutional layer, we apply a dropout layer [17] with rate 0.2.

4.2 Transition Block

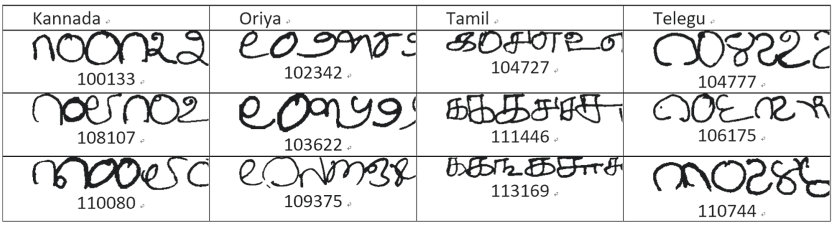
The feature maps pass through a dense block will keep the size the same. But it is important to reduce the feature map size in a convolutional network, so we apply a transition block between two dense blocks to decrease the size. A typical transition block consists of a batch normalization layer, a 1*1 convolutional layer and an average pooling layer with kernel 2*2. The dropout layer has dropout rate to 0.2.

4.3 Residual Connections

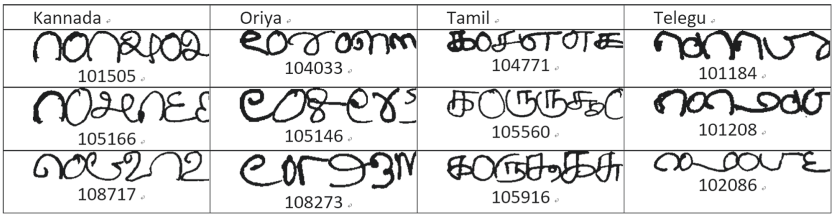
There are two main differences between DenseNet [12] and ResNet [14]. In DenseNet, the connections between convolutional layers are densely and the way to combine feature maps is concatenation. We enhance these advantages in this paper. In transition block there is an average pooling layer to reduce the feature map size, in order to retain more information, we add max pooling residual connections, as shown in Fig. 4. And we follow the way in DenseNet, we concatenate the two branches features to feed into next layer.



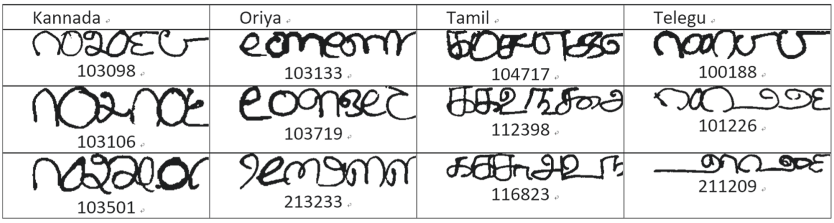
(a) Dataset-1



(b) Dataset-2



(c) Dataset-3



(d) Dataset-4

Fig. 3. Samples of different datasets.

4.4 Dimension Adjustment

The output of convolutional layer always has three dimensions. However, the CTC we apply in our network requires the input data with two dimensions. The output of dense block is always a 3-D tensor, i.e., the *number*, *height* and *width* of feature maps (4-D if we consider the $batch - size$ dimension). First we flatten the 3-D feature tensor to 2-D by expanding on the *number* dimension, then we apply a column-wise fully connected layer to reduce the *height* dimension to the assigned value. With these actions, we can generate the suitable input for the following CTC output layer.

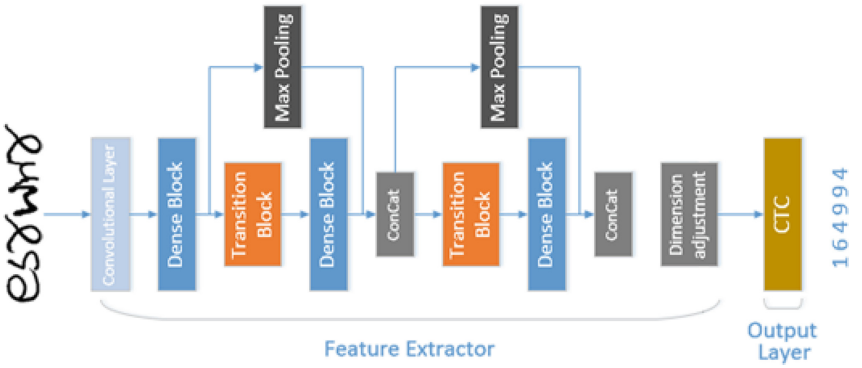


Fig. 4. The structure of the proposed network.

4.5 CTC Output Layer

Connectionist temporal classification [13] is a kind of output layer with two main functions, calculating the loss at training phase and generating the prediction results at testing phase.

For a string recognition task, the labels are drawn from a set A (in this paper, A is the ten digits). With an extra label named blank, we get a new set $A^{'} = A \bigcup$ blank, which is used in reality. The input of CTC is a sequence $y = y_1, ..., y_T$, where T is the sequence length. The corresponding label denotes as I over A. Each y_i is a probability distribution on the set $A^{'}$. We define a many-to-one function $F : A_T => A_{<=T}$ to resume the repeated labels and blanks. For example $F(11-6-49-999-44-----) = 164994$. ($-$ indicates the 'blank' label).

Let $S = (X, I)$ is the training set, where X is the training image and I is the ground truth. So the CTC loss is calculated as:

$$\mathcal{O}(S) = -\sum_{(x,I)\in S} \log p(I|y) \tag{1}$$

where $p(I/y)$ is the conditional probability defined as the sum of probabilities of all predictions that are mapped by F onto I, I is the prediction result. Therefore, the network can be end-to-end trained on pairs of images and sequences, without the procedure of manually labeling individual components in training images.

5 Result and Discussions

Before going to present the results, we provide here parameters and system information. We apply ADADELTA to update the parameters. In all experiments, we train the network with 50 epochs. The hyper-parameters of three dense blocks are the same with the growth rate to 8 and the number of convolution layer to 8. Our experiments are performed on a Super-micro server with the GPU NVIDIA TITAN X. The software is the Caffe [20] framework with cuDNN V5 accelerated.

5.1 Global Recognition Results

Overall accuracies on 16 datasets (four datasets for each of the four scripts) obtained from the experiments for Kannada, Oriya, Tamil and Telugu scripts are shown in Table 1. Here both digit level as well as string level accuracies are presented here. From the table it can be seen that maximum accuracy for digit level is obtained from Telugu (98.90%) and it is for dataset 1. Minimum accuracy for digit level is obtained from Kannada (96.42%) and it is for dataset 4. Similarly, that maximum accuracy for string level is obtained from Tamil (91.78%) and it is for dataset 1. Minimum accuracy for string level is obtained from Kannada (75.94%) and it is for dataset 4. It can be seen that dataset 4 has relatively lower accuracy than other datasets and this is because dataset 4 is the most complex dataset having many types of touching.

Table 1. Digit level and string level accuracies of four scripts on different datasets.

Datasets & mode		Scripts			
		Kannada	Oriya	Tamil	Telugu
Dataset-1	Digit level Accuracy	97.43	98.00	98.87	98.90
	String level Accuracy	82.70	85.08	91.78	91.39
Dataset-2	Digit level Accuracy	97.57	97.20	98.69	98.39
	String level Accuracy	83.52	80.60	90.52	88.17
Dataset-3	Digit level Accuracy	97.26	96.97	98.33	98.55
	String level Accuracy	81.51	78.50	88.30	89.01
Dataset-4	Digit level Accuracy	96.42	96.64	98.34	98.21
	String level Accuracy	75.94	76.84	88.59	86.52

5.2 Confusing Numeral Pair Computation

We also noted the main confusing numeral pair of different scripts considered here and we observed that main reason of such confusion is shape similarity. Four confusing matrices for the four scripts on dataset-1 are presented in the Tables 2, 3, 4 and 5 for Kannada, Oriya, Tamil and Telugu scripts, respectively.

In Kannada main confusing numeral pair is numeral six and numeral seven. They confuse about 2.37% cases. Next pair of confusion in Kannada is numeral three and numeral seven with confusion rate 2.35%. For Oriya, maximum confusion is between is numerals two and seven they confuse about 3.04% cases. It can be seen that from Table 4, for Tamil, maximum confusion is between numerals four and six and they confuse about 1.19% cases. Similarly, it can be seen that from Table 5 for Telugu that numeral nine and six have maximum confusion it is 1.74%.

Table 2. Confusion matrix for Kannada.

Digit	Recognized as									
	0	1	2	3	4	5	6	7	8	9
0	98.31	1.01	0.22	0.15	0.01	0.17	0.0	0.0	0.12	0.01
1	0.01	99.89	0.02	0.05	0.0	0.02	0.0	0.0	0.0	0.0
2	0.01	0.0	99.46	0.19	0.01	0.14	0.0	0.01	0.03	0.14
3	0.0	0.05	0.01	99.35	0.01	0.44	0.0	0.03	0.0	0.1
4	0.0	0.02	0.0	0.2	98.42	1.09	0.07	0.03	0.07	0.09
5	0.0	0.04	0.49	1.19	0.21	97.96	0.0	0.01	0.06	0.05
6	0.01	0.0	0.04	0.26	0.43	0.07	94.59	2.37	2.02	0.19
7	0.03	0.03	0.41	2.35	0.07	0.06	0.27	95.97	0.76	0.06
8	0.23	0.04	0.11	0.12	0.01	0.15	0.02	0.0	98.91	0.42
9	0.0	0.1	0.0	0.02	0.02	0.0	0.42	0.01	0.17	99.26

Table 3. Confusion matrix for Oriya.

Digit	Recognized as									
	0	1	2	3	4	5	6	7	8	9
0	99.15	0.02	0.02	0.08	0.02	0.0	0.29	0.03	0.05	0.35
1	0.06	99.2	0.0	0.03	0.0	0.28	0.0	0.11	0.31	0.0
2	0.07	0.04	96.15	0.2	0.06	0.0	0.37	3.04	0.0	0.06
3	0.01	0.0	0.0	98.73	0.0	0.0	0.27	0.93	0.03	0.01
4	0.17	0.0	0.0	0.12	99.45	0.09	0.0	0.1	0.04	0.02
5	0.0	0.3	0.0	0.57	0.4	98.51	0.0	0.02	0.0	0.2
6	0.29	0.29	0.35	0.2	0.02	0.0	97.15	1.66	0.01	0.03
7	0.0	0.0	1.22	0.5	0.02	0.0	0.09	98.07	0.0	0.08
8	0.0	0.01	0.0	0.13	0.0	0.0	0.0	0.0	99.85	0.0
9	0.14	0.04	0.01	0.55	0.05	0.14	0.08	0.02	0.16	98.81

5.3 Erroneous Results

To get the idea about the digit-string samples where our system provides erroneous results we provide some examples in Fig. 5. Here four samples are given, In the first samples the actual string 1000018082 is recognized as 100018082. The first two samples show the mistake about losing one digit. In the second samples the actual string 10014801 is recognized as 1001480. In the third and fourth samples, there are some wrong predictions and this is mainly because of their shape similarity.

Table 4. Confusion matrix for Tamil.

Digit	Recognized as									
	0	1	2	3	4	5	6	7	8	9
0	99.55	0.03	0.0	0.0	0.01	0.0	0.0	0.37	0.0	0.03
1	0.0	99.73	0.01	0.0	0.01	0.0	0.14	0.0	0.0	0.1
2	0.04	0.02	98.66	0.31	0.0	0.08	0.02	0.01	0.86	0.0
3	0.0	0.06	0.03	99.77	0.0	0.11	0.0	0.04	0.0	0.0
4	0.0	0.53	0.0	0.0	98.02	0.0	1.19	0.03	0.23	0.0
5	0.0	0.01	0.0	0.0	0.0	99.94	0.03	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.02	0.01	99.74	0.05	0.0	0.18
7	0.0	0.0	0.0	0.01	0.0	0.01	0.18	99.73	0.05	0.0
8	0.01	0.0	0.0	0.0	0.01	0.0	0.11	0.0	99.88	0.0
9	0.1	0.02	0.01	0.21	0.02	0.34	0.7	0.02	0.0	98.59

Table 5. Confusion matrix for Telugu.

Digit	Recognized as									
	0	1	2	3	4	5	6	7	8	9
0	99.19	0.77	0.01	0.0	0.02	0.01	0.0	0.01	0.0	0.0
1	0.08	99.84	0.0	0.02	0.0	0.0	0.0	0.05	0.0	0.0
2	0.07	0.0	99.68	0.12	0.11	0.0	0.01	0.0	0.0	0.0
3	0.02	0.03	0.01	99.83	0.07	0.0	0.0	0.03	0.0	0.0
4	0.0	0.0	0.0	0.0	99.77	0.22	0.0	0.0	0.0	0.0
5	0.01	0.03	0.01	0.02	0.54	99.39	0.01	0.01	0.0	0.0
6	0.06	0.0	0.0	0.01	0.03	0.01	98.65	0.34	0.0	0.89
7	0.0	0.0	0.01	0.01	0.02	0.0	0.0	99.94	0.0	0.0
8	0.09	0.0	0.0	0.02	0.2	0.0	0.0	0.0	99.67	0.0
9	0.01	0.01	0.0	0.01	0.08	0.0	1.74	0.01	0.22	97.91

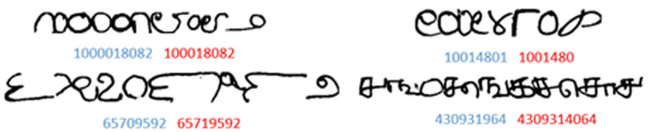


Fig. 5. Examples of some similar shaped numerals.

6 Conclusion

In this paper we apply a CNN-CTC architecture for four handwritten Indian script numeral string recognition. We also develop datasets of digit string for these four scripts. Each dataset has 20000 numeral string samples for training and 30000 samples for testing. As there is no such dataset available, it will be helpful to the community. Moreover, to the best of our knowledge there is no work on digit-string recognition of these four scripts, and hence this is the first work on work on these scripts.

References

1. Plamondon, R., Srihari, S.N.: On-line and off-line handwritten recognition: a comprehensive survey. IEEE Trans. PAMI **22**, 62–84 (2000)
2. Pal, U., Chaudhuri, B.: Indian script character recognition: a survey. Pattern Recogn. **37**, 1887–1899 (2004)
3. Bhowmick, T., et al.: An HMM based recognition scheme for handwritten Oriya numerals. In: Proceedings of the 9th International conference on Information Technology, pp. 105–110 (2006)
4. Sharma, N., Pal, U., Kimura, F.: Recognition of handwritten Kannada numerals. In: Proceedings of the 9th International Conference on Information Technology, pp. 133–136 (2006)
5. Hanmandlu, M., Ramana Murthy, O.: Fuzzy model based recognition of handwritten numerals. Pattern Recogn. **40**, 1840–1854 (2007)
6. Wen, Y., Lu, Y., Shi, P.: Handwritten Bangla numeral recognition system and its appli-cation to postal automation. Pattern Recogn. **40**, 99–107 (2007)
7. Bajaj, R., Dey, L., Chaudhury, S.: Devnagari numeral recognition by combining deci-sion of multiple connectionist classifiers. Sadhana **27**, 59–72 (2002)
8. Kumar, S., Singh, C.: A study of Zernike moments and its use in Devnagari handwrit-ten character recognition. In: Proceedings of the International conference on Cognition and Recognition, pp. 514–520 (2005)
9. Bhattacharya, U., et al.: Neural combination of ANN and HMM for handwritten Devnagari numeral recognition. In: Proceedings of the 10th International Workshop on Frontiers of Handwriting Recognition, pp. 613–618 (2006)
10. Otsu, N.: A Threshold selection method from grey level histogram. IEEE Trans. SMC **9**, 62–66 (1979)
11. Kimura, F., et al.: Modified quadratic discriminant function and the application to Chinese character recognition. IEEE Trans. PAMI **9**, 149–153 (1987)
12. Huang, G., Liu, Z., Weinberger, K., Maaten, L.: Densely connected convolutional networks (2016). arXiv preprint arXiv:1608.06993
13. Graves, A., Fernndez, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd International Conference on Machine learning, pp. 369–376 (2006)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

15. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Proceedings of International Conference on Machine Learning, pp. 448–456 (2015)
16. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks, In: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, pp. 315–323 (2011)
17. Hinton, G., et al.: Improving neural networks by preventing co-adaptation of feature detectors (2012). arXiv preprint arXiv:1207.0580
18. Pal, U., Roy, K., Kimura, F.: Bangla handwritten pin code string recognition for indian postal automation. In: Proceedings of International Conference on Frontiers in Handwriting Recognition, pp. 290–295 (2008)
19. Pal, U., Roy, K., Kimura, F., Indian multi-script full pincode string recognition for postal automation, In: Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR), pp. 456–460 (2009)
20. Jia, Y., et al.: Caffe: convolutional architecture for fast fea-ture embedding (2014). arXiv preprint arXiv:1408.5093