Optical Character Recognition for Handwritten Marathi digits using Convolutional Neural Networks

Tushar Badgu, Prasad Bhagwat, Venkatesh Lokare
Department of Computer Engineering, PICT
Savitribai Phule Pune University
tushar.badgu@gmail.com
prasad.bhagwat14k@gmail.com
venky177@gmail.com

***Abstract*—This paper imparts knowledge about optical character recognition for handwritten Marathi digits. Data is collected from over 200 people and expanded to over 20000 digits using image pre-processing mechanisms such as scaling, skewing, etc. A convolutional neural network (CNN) is proposed for classifying 0-9 in Marathi over this large self-built Marathi dataset. Various parallelization mechanisms such as NVIDIA GPUs have been used to reduce the training time and gain an accuracy of 99.73% .This paper provides a streamlined approach for collection of data, its expansion to form a dataset and further classification of the dataset using a CNN, comparing it with other traditional methods in the results section we provide substantial evidence as to why this method is more robust than its competitors.**

***Index Terms*—Optical Character Recognition, Image Pre-processing, Convolutional Neural Network, Feature Extraction.**

1. Introduction

Marathi is the official language in Maharashtra and co-official language in Goa - states of Western India, and is one of the 23 official languages of India. Over 50 million people of Maharashtra speak Marathi. Marathi ranks 19th in the list of most spoken languages in the world. Marathi has the fourth largest number of native speakers in India [1]. Though it has rich culture and literature, and is a very widely spoken language, hardly any significant work has been done for the identification of Marathi optical characters. The numerals in Indian languages are based on sharp curves and hardly any straight lines are used. Figure 1 is a set of Marathi numerals.

There are many pieces of work towards handwritten recognition of Roman, Japanese, Chinese and Arabic scripts, and various approaches have been proposed by the researchers towards handwritten character recognition [2-4]. Also there are many pieces of work that have been done towards the recognition of Indian printed characters [5-9] and at present optical character recognition systems are commercially available for some of the printed Indian scripts. First research report on handwritten Devnagari characters was published in 1977 [10] and not much research work is done after that. Some research work are available towards Devnagari numeral recognition [11-13]. But none of them uses CNN for digit recognition. As with the CNN accuracy is always better.

This paper addresses the problem of handwritten Marathi numeral recognition. Gujarati numeral recognition requires smoothing, thinning, skew detection and correction, and normalization process is performed as pre-process. Further, profiles are used for feature extraction and convolutional neural network (CNN) is suggested for the classification. The organization of this paper is as follows:Pre-processing techniques are described in Section 2. Section 3 is for the explanation of feature extraction. Section 4 describes the suggested convolutional neural network. Section 5 describes the training of the network. The results are explained in Section 6. Lastly conclusion is described in Section 7.

० १ २ ३ ४ ५ ६ ७ ८ ९

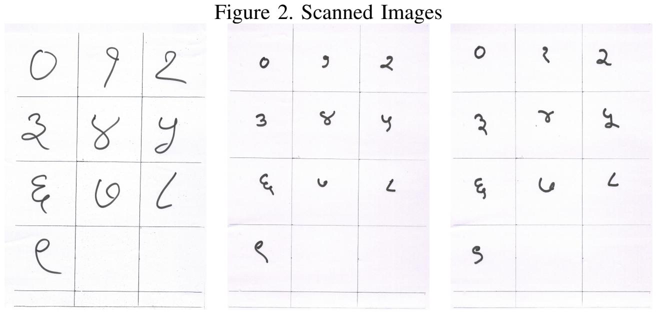
Figure 1. Marathi digits 0-9

2. Pre-Processing

For developing a system to identify Marathi handwritten digits, we have collected numerals 09 written in Marathi from 300 different people of various background and different genders. These numerals were written on a paper having 12 boxes of size 7 cm x 7 cm and each individual wrote the numerals in those boxes. These papers were scanned in 300 dpi by a flatbed scanner. In Figure 2 the scanned images are illustrated which show the need for pre-processing of the images before processing them for the classification.

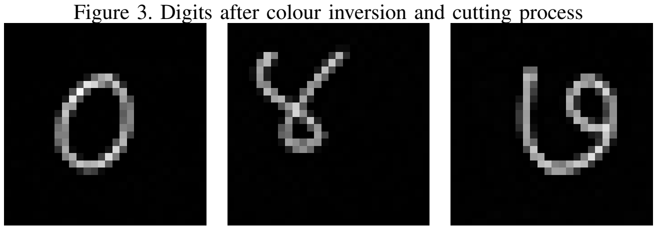
The first pre-process which is performed is inverting colours of the scanned images so that the background becomes black and the digits are white in colour. The next step is extracting individual numerals from those images. For this we cut each box of the scanned image for individual digit and adjusted

Figure 2. Scanned Images



the size of the resulting image to 28 x 28 pixels. Some of the results are as shown in Figure 3.

Figure 3. Digits after colour inversion and cutting process



The various preprocessing steps helped us increase the raw training data from 200x10 to 2000x10. We used transformation matrices, to aid in conversion of the raw image to a new undocumented training data image.

The various transformation performed were:

- Translation: A function which moves a point for a predetermined distance.
  The Translation matrix is as follows:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ V_x & V_y & 1 \end{bmatrix}$$

  Here V_x,V_y are displacement values in the X, Y directions respectively.

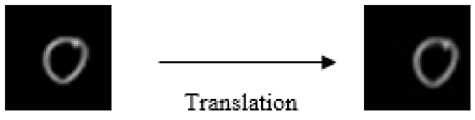


Figure 4. Translation Example

- Scaling: A function which enlarges the dimensions of an image.
  The Scaling matrix is as follows:

$$\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  Here S_x,S_y are scaling values in the X, Y directions respectively.

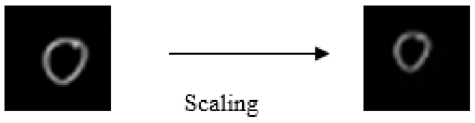


Figure 5. Scaling Example

- Shearing: A function which displaces each point in fixed direction, proportional to its distance from a line that is parallel to that direction.
  The Shearing matrix is as follows:

$$\begin{bmatrix} 1 & A & 0 \\ B & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  Here A,B are shearing values in the X, Y directions respectively.

Figure 6. Shearing Example

These functions were convoluted with the training images of size 28x28, with various permutations and combinations to convert 1 raw image to 9 similar convoluted images. We used various image processing tools such as OpenCV and Octave to help in these endeavors. Eventually increasing our dataset ten folds in size.

We faced various challenges such as, transformed image leaving the 28x28 boundary, and also the salt and pepper error caused during transformation of images. We tackled the first by centering the images or discarding the disproportionate image completely. The salt and pepper error was removed using inbuilt kernels in OpenCV.

Finally these images, 18303 in number, of size 28x28 were streamlined, converting a 28x28 matrix to a 784 vector. These new vectors were stacked over each other, forming a 784x18303 matrix which we would further be used for training.

3. Feature Extraction

The images used for optical character recognition have the property of being stationary i.e. the statistics of one part of image is same as any other part. This property is exploited by CNN in which the features that we learn from one part of the image can be applied to other parts of the image.

A Convolutional Neural Network can be used for feature extraction as well as classification of the images. The features extracted can be used to train a classifier such as SVM and then perform classification of the input data. Or else the same network used for feature extraction can be made to classify the input data by adding an output layer. Based on the scores of the output layer neurons the predicted label/class of the input image can be identified.

Feature extraction in CNN is done using the convolve operation. Many predefined filters exists which can be applied on the images to extract the features such as edges, gradients, textures, etc. But using hand-coded features for classification task results in less accuracy as compared to the features that are learnt using the training images. So,a broad view of CNN would be to learn a set of filters during training of the network and then later use those filters as features to predict the labels of the test images. A CNN works on 3-dimensional volume as compared to a fully connected network. Feature extraction in CNN is based on three factors used in the convolutional layer. Let us discuss about these factors to get an intuition of how a CNN works.

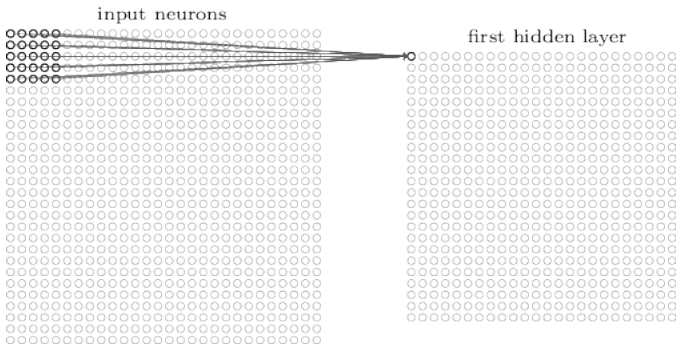


Figure 7. Local Receptive Fields

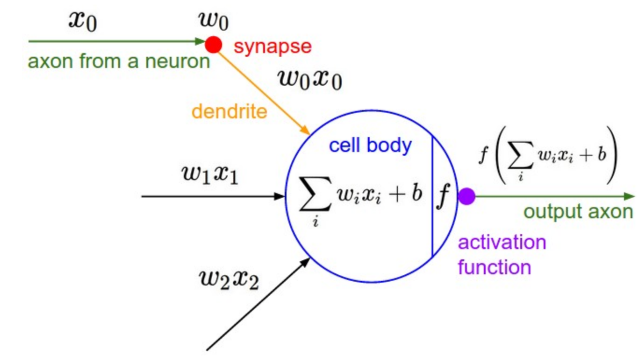


Figure 8. Activation of single neuron using shared weights and shared bias

3.1. Local Connectivity

According to the stationary property of images, it would be impractical to connect a neuron to all the neurons in the previous layer. Hence, we will connect a neuron to only a local region of the previous layer of neurons. The spatial extent of this loacl region is decided by a Hyper-parameter called the Receptive field. The extent of the connectivity along the depth axis is always equal to the depth of the input volume.

For Example, for a input volume of size [20*20*3] and receptive field of size [5*5] , each neuron in the convolutional layer will have weights to a [5*5*3] region in the input volume.

3.2. Parameter Sharing

Each hidden neuron in a Conv Layer has a weight matrix whose size is determined by the local receptive field and a bias value as well. CNN uses parameter sharing to reduce the no. of parameters in such a way that all the neurons in one particular layer of the Conv layers uses the same set of weights and bias. This makes sure that one layer will detect only one feature throughout the image. For this reason the map from input layer to the hidden layer is called a feature map.

3.3. Spatial Arrangement

Spatial Arrangement is used to determine the number of neurons in the Conv Layer and their arrangement. Three hyperparameters control the spatial arrangement which are depth, stride and zero-padding.

Depth of the output volume is a hyperparameter that determines the number of neurons that connect to the same region of the input volume. Simply, it determines the number of feature maps to be learned. Each feature map detects a particular feature along the input raw image which may be presence of various oriented edges, or blobs of color.

Stride determines the length/number of neurons by which the local region must be moved for the next neurons activation. Suppose if the stride is 1, then we will allocate a new column of neurons along the depth to spatial positions only 1 unit apart. This will eventually lead to high overlapping of the local regions between the columns, and also lead to generation of large output volumes. Conversely, if we use higher strides then the local regions will overlap less and the resulting output volume will have smaller dimensions spatially.

Next parameter is zero-padding which is done in some cases to preserve the dimensional volume along the network so as to stack multiple convolutional layers on top of each other. The size of this zero-padding is a hyperparameter. It is mostly used when we want to preserve the dimension of the input volume.

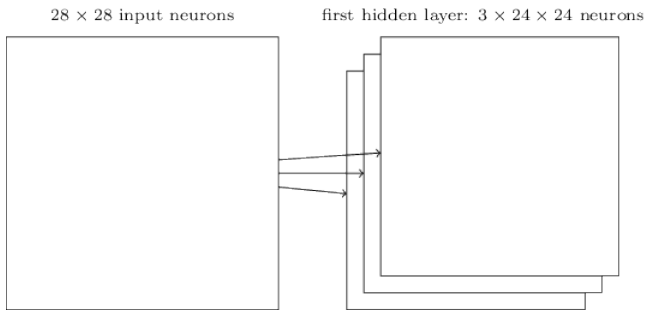


Figure 9. Spatial arrangement and Feature maps

We can compute the spatial size of the output volume as a function of the input volume size (W), the receptive field size of the Conv Layer neurons (F), the stride with which they are applied (S), and the amount of zero padding used (P) on the border. The formula to calculate the number of neurons on the basis of hyperparameters is (WF+2P)/S+1.

4. Convolutional Neural Network

The problem of digit recognition from 2-D images is typically very ill-posed i.e , there are many models which fit the training points well but do not generalize well to unseen images. In other words, the model might fail for slight variation in the images as compared to those seen in the training set. Also, the neural network with multiple layers which are fully-connected when trained on the input images are not invariant to transformation and local deformation. All such factors led to the development of an architecture specialized to classify images i.e. Convolutional Neural Network (CNN).

Convolutional Neural Networks are biologically-inspired variants of MLPs and are more focussed towards the visual part of the brain. A CNN is able to achieve some degree of shift and deformation invariance with the help of three concepts: local receptive fields, shared weights, and pooling. Before starting with the discussion of the concepts above lets understand the input which is fed to the network.

The CNN developed is been trained to recognize Marathi numeral digits from the images. Each image contains a single Marathi numeral digit from 0-9 and the resolution is 28*28. So, a total of 784 pixel values define one point in the input set.

In traditional neural networks each neuron in one layer is connected to all the neurons in the next layer due to this the neural network is unable to capture the local features in an image. Here comes the concept of local receptive fields. In a CNN, a neuron in the hidden layer is connected to a small region of input neurons which makes the connections in the network responsive to small, localized regions in the image. For example, we can consider a 5*5 region of the input image mapped to a single neuron in the hidden layer. That region in the input image is called the local receptive field for the hidden neuron. The local receptive field is slide by amount of stride length to the right to connect to the second hidden neuron. So, for a 28*28 input image and stride length of 1 we obtain a 24*24 neurons in the hidden layer.

We know that a neuron in the hidden layer has a bias and the connections to the neurons have a weight. The variation in CNN is that all the local receptive fields responsible for the formation of a hidden layer share a same set of weights and bias. So, for a 5*5 local receptive field there will be a 5*5 shared weights and a shared bias used by all the neurons in the hidden layer. The importance of using shared weights and bias is that all the neurons in the hidden layer detect exactly the same features but at different locations within the image. The map from the input layer to the hidden layer is called a feature map. But for the purpose of image recognition, a single feature map is not enough, so we develop multiple feature maps each recognizing a different feature.

The activation of j,k the hidden neuron is obtained as follows :

$$\sigma\left(b + \sum_{l=0}^{4}\sum_{m=0}^{4} w_{l,m} a_{j+l,k+m}\right). \tag{1}$$

Here, is the neural activation function, b is the shared value of bias. $w_{l,m}$ is a 5*5 array of shared weights and $a_{x,y}$ denotes the input activation at position x, y.

A CNN also has a pooling layer stacked just after the convolutional layer. The basic aim of using pooling layer is to simplify the information in the output of convolutional layer. A pooling layer takes each feature map as input and produces a compact feature map without the loss of information. Mostly, the pooling layer summarizes 2*2 regions in the feature maps. The commonly used pooling procedure is Max-pooling in which the pooling unit simply outputs the maximum activation in the 2*2 input region. Another approach is L1 pooling in which square root of the sun of squares of the activations in the 2*2 region is produced as the output activation.

5. Training the Network

As mentioned above 18310 images were used to train the CNN. Each image was streamlined into a 784 vector as follows:
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

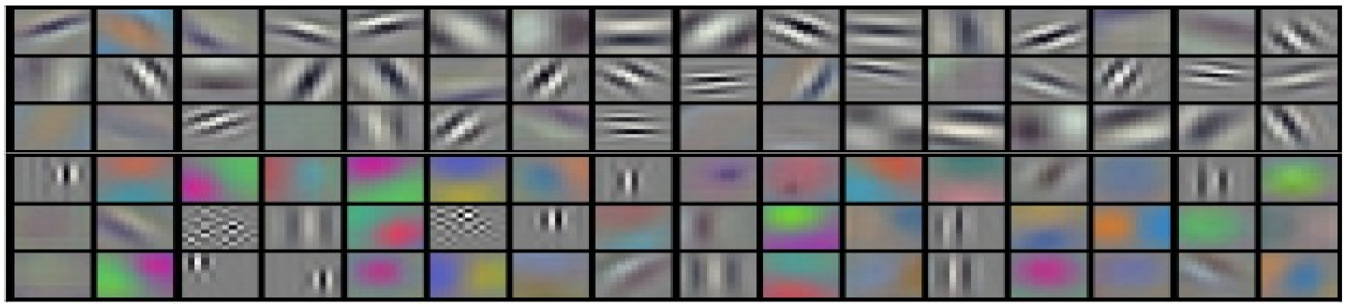


Figure 10. Example filters learned by Krizhevsky et al.

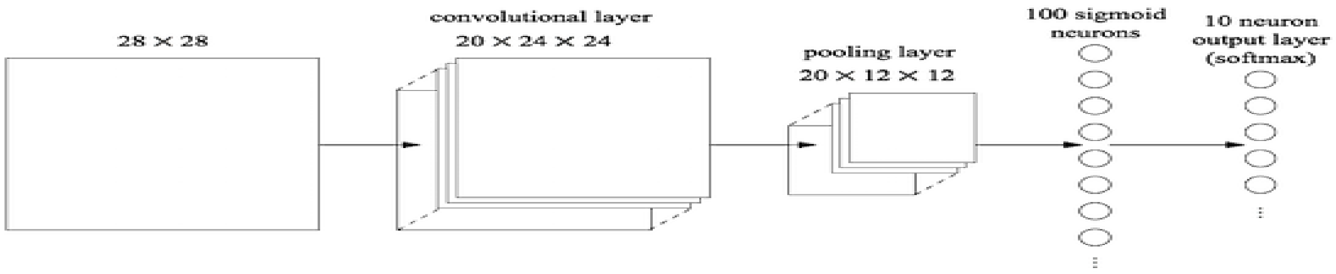


Figure 11. Convolutional Neural Network Architecture

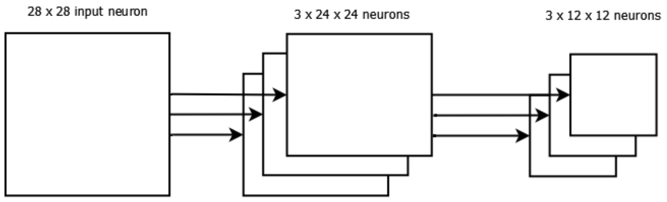


Figure 12. Conv Layer

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,35,90,88,101,97,10,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,64,108,118,102,90,99,148,135,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,93,127,85,4,0,
0,0,43,158,82,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,57,141,
43,0,0,0,0,0,0,132,98,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
43,101,0,0,0,0,0,0,4,122,70,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,64,54,0,0,0,0,0,67,98,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,40,85,4,0,0,0,32,93,40,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,88,110,67,50,78,88,32,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,82,113,115,85,10,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0.

Vector representation for the number 0 is shown above. The greyscale image is having values 0 for black intensity pixels and 255 for white intensity.

These vectors are then stacked to form an 18310x784 matrix which was fed to the CNN. The processor used was Intel i7 with 8 GB Ram and graphic card used was Nvidia Tesla K40 with 2880 cuda cores. The theano framework for python was used for creating the CNN. Training required 40 minutes. 18310 amount of data was used as cross verification and 5500 was used as test dataset for actual result calculation.

6. Results

This network was trained for 10 sets of each digit from 0-9 wherein each set had 1281 samples of that particlular digit. So, in total the number the samples used for training were 12810. And for testing of the network again we used 550 samples of each digit which were seprate from the training set. So the test was conducted on a total number of 5500 samples.

We obtained an overall test accuracy of 99.73% for the Convolutional Neural Network trained for the purpose of Optical Character Recognition. The test accuracy for individual digits from 0-9 has been given in TABLE 1.

7. Conclusion and Future Work

Thus using convolutional neural networks we could see a higher accuracy in the classification of Marathi digits. This

TABLE 1. PERFORMANCE OF NETWORK FOR TEST SET

Digits	Accuracy(%)
0	98.0
1	98.6
2	96.8
3	97.8
4	98.5
5	96.0
6	97.2
7	99.5
8	99.0
9	98.6

goes to prove the importance of vast applications of deep neural nets in the field of Artificial Intelligence. With the growth of faster processing units, the training time over large datasets has been considerable reduced, exploiting this fact further applications using deep neural nets can be developed and trained to gain substantial improvements in the supervised learning domain.

8. Acknowledgements

We would like to take this opportunity to thank our guide Prof. D.T.Mane, Pune Institute of Computer Technology for giving us all the help and guidance we needed. We are really grateful to him for his kind support throughout the analysis and research work. We are also grateful to other staff members of the Department for giving important suggestions.

We are thankful to PICT Computer Department, for providing us ultimate references and resources such as the Nvidia Gpu enabled machines, which helps us lot to do research and study topics related to the paper.

References

[1] *Abstract of Language Strength in India: 2001 Census*. Censusindia.gov.in. Archived from the original on 10 February 2013. Retrieved 2013-05-09.

[2] R. Plamondon and S. N. Srihari, *On-Line and off-line handwritten recognition: A comprehensive survey*, IEEE Trans on PAMI, Vol.22, pp.62-84, 2000.

[3] Z.Chin and H. Yan, *A handwritten character recognition using self-organizing maps and fuzzy rules*, Pattern Recognition, Vol.22, pp. 923-937, 2000.

[4] K.Kim and S.Y. Bang, *A handwritten character classification using tolerant Rough set*, IEEE Trans. on PAMI, Vol.22, pp.923-937, 2000.

[5] U. Pal and B.B. Chaudhuri, *Indian script character recognition: A Survey*, Pattern Recognition, Vol. 37, pp. 1887-1899, 2004.

[6] B. B. Chaudhuri and U. Pal, *A complete printed Bangla OCR system*, Pattern Recognition, Vol. 31, pp. 531-549, 1998.

[7] K. G. Aparna, A. G. Ramakrishnan, *A Complete Tamil Optical Character Recognition System*, In Proc. 5th Intl workshop on DAS, pp. 53-57, 2002.

[8] A. Negi, C Bhagvati, B. Krishna, *An OCR System for Telugu*, In Proc. 6th ICDAR, pp.1110-1114, 2001.

[9] V. Bansal and R. M. K. Sinha, *A complete OCR for printed Hindi text in Devnagari script*, In Proc. 6th ICDAR, pp. 800-804, 2001.

[10] I. K. Sethi and B. Chatterjee, *Machine Recognition of constrained Hand-printed Devnagari*, Pattern Recognition, Vol. 9, pp. 69-75, 1977.

[11] M. Hanmandlu and O.V. Ramana Murthy, *Fuzzy Model Based Recognition of Handwritten Hindi Numerals*, In Proc. Intl. Conf. on Cognition and Recognition, pp. 490-496, 2005.

[12] R. Bajaj, L. Dey, and S. Chaudhury, *Devnagari numeral recognition by combining decision of multiple connectionist classifiers*, Sadhana, Vol.27, pp.-59-72, 2002.

[13] U. Bhattacharya, S. K .Parui, B. Shaw, K. Bhattacharya, *Neural combination of ANN and HMM for handwritten Devnagari Numeral Recognition*, In Proc. 10th IWFHR, pp.613-618, 2006.