**Cursive Handwritten Text Segmentation & Recognition**

Introduction:

Handwriting recognition is one of the very challenging and difficult problems, Recently, a number of papers have been published with research detailing new techniques for the classification of handwritten numerals and words. In most of the research text recognition carried out by passing each small part of segmented text to neural network. This technique relies on the confidence values for verification of correctness which is very slow & difficult to implement.

In this experiment we proposed very simple and unique algorithms for text recognition which mainly focus on the word segmentation. The algorithm involves the preprocessing, segmentation and recognition steps.

Slant Correction:

To make the letter recognition more efficient we corrected the slantness of image. For slant correction we first calculate the angle for maximum lines with respect to vertical line using ‘Hough Lines method’. After finding the angle, image is rotated with same angle in opposite direction around the baseline of text.

|  |  |
| --- | --- |
| C:\Users\493087\Documents\PythonProj\data\f.png | C:\Users\493087\Documents\PythonProj\data\r2.jpg |
| Fig. Slant Image | Fig. Deslant Image |

Line Segmentation:

Considering the text lines are horizontal in position, the text lines are separated out by extracting the row wise consecutive black pixel existence.

|  |
| --- |
| C:\Users\493087\Documents\PythonProj\data\Result\LetterSeg\0str.jpeg |
| C:\Users\493087\Documents\PythonProj\data\Result\LetterSeg\1str.jpeg |
| C:\Users\493087\Documents\PythonProj\data\Result\LetterSeg\2str.jpeg |
| Fig: Text Lines |

Word Segmentation:

After extracting the text lines, the image is dilated to join the distant neighboring pixel or to remove the small space between two letters. After that the we tried to use contour tracing to accurately segment the intersecting components. Then bounding box is applied on contours to extract words. These word images are then further sent for preprocessing.

|  |
| --- |
| C:\Users\493087\Documents\PythonProj\data\Result\LetterSeg\10cont.jpeg |
| Fig: Word Separation using contour |

Preprocessing:

The first step to any document analysis is pre-processing. First the sample image is converted to gray scale image. A binary image is generated out of this grey scale image and one bit binary noise is removed by using erosion. The subsequent steps are all for normalizing the data to common axis. In normalization the various sized images are resized to fixed size image keeping aspect ratio intact.

Top Contour Point Extraction:

The top contour points are extracted by taking first black pixel of each column from top.

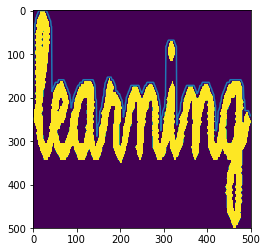


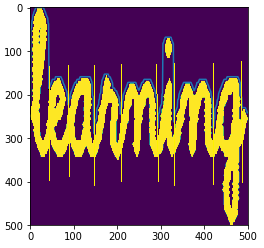
Fig: Graph plot in blue over the top contour points

Character segmentation:

We have proposed two approaches for the character segmentation

1. Local maxima & Minima Technique:

In this we traverse through each top contour points from left to right. Then the vertical distance of each contour point with its next contour point is measured. If this distance is more than certain threshold value, then we consider this region as possible segment line.



1. Middle Cut Line method:

In this method, the upper contour points and lower contours points are extracted. And then mean value is calculated for both upper & lower contour points. This mean value gives us the approximate upper base line & lower base line. Considering these two lines we find the mid line and cut the word image from this mid line.

After cutting the image from mid line we traverse through this line and find the complete blank spaces. This region can be considered as segment point. The segment points are then applied to original word image for segmentation.

|  |  |
| --- | --- |
|  |  |
| Fig: Cut Image from middle of Upper & lower baseline and segmented | |

Over Segmentation removal:

In the segmentation process, due to bad quality images, words are getting oversegmented. We tried to minimize the oversegmentation problem by thresholding. If the distance between two segment lines are less than the threshold value, then that segment line will be rolled out and consider its next segment line.

C:\Users\493087\Documents\PythonProj\data\Result\66.jpg C:\Users\493087\Documents\PythonProj\data\Result\67.jpg C:\Users\493087\Documents\PythonProj\data\Result\68.jpg C:\Users\493087\Documents\PythonProj\data\Result\69.jpg  C:\Users\493087\Documents\PythonProj\data\Result\71.jpg  C:\Users\493087\Documents\PythonProj\data\Result\73.jpg

Fig. Segmented letters

Image Standardization:

As we are going to evaluate the segmented letter images on standardize trained image model which is of fixed size 28\*28 pixel, we resize the images with same size including padding of 4 pixels on each side.

We cropped the various sized input images to cut the extra padding on each side and then resize to 20\*20 pixel.

C:\Users\493087\Documents\PythonProj\data\Result\Image\66.jpg C:\Users\493087\Documents\PythonProj\data\Result\Image\67.jpg C:\Users\493087\Documents\PythonProj\data\Result\Image\68.jpg C:\Users\493087\Documents\PythonProj\data\Result\Image\69.jpg C:\Users\493087\Documents\PythonProj\data\Result\Image\70.jpg C:\Users\493087\Documents\PythonProj\data\Result\Image\71.jpg C:\Users\493087\Documents\PythonProj\data\Result\Image\72.jpg C:\Users\493087\Documents\PythonProj\data\Result\Image\73.jpg

Fig: Segmented letters in 28\*28-pixel size

Character Recognition:

In this application we propose the use of Convolution Neural Networks for recognition of characters which utilizes the unsupervised learning technique for training. We have created CNN multilayered model using KERAS backed by Tensorflow. For training purpose, we have used MNIST character dataset. The dataset consists of over 1L labeled images of handwritten letters. The images are in 28\*28-pixel format.

We trained the model on 70% of data and tested on 30% of data. The measured accuracy got from the CNN model is around 90%.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_1 (Conv2D) (None, 32, 26, 26) 320

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_2 (Conv2D) (None, 32, 26, 26) 9248

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_1 (Activation) (None, 32, 26, 26) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_1 (MaxPooling2 (None, 32, 13, 13) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_3 (Conv2D) (None, 64, 13, 13) 18496

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_2 (Activation) (None, 64, 13, 13) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_4 (Conv2D) (None, 64, 13, 13) 36928

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_3 (Activation) (None, 64, 13, 13) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_2 (MaxPooling2 (None, 64, 6, 6) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_5 (Conv2D) (None, 32, 6, 6) 18464

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_4 (Activation) (None, 32, 6, 6) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_6 (Conv2D) (None, 32, 6, 6) 9248

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_5 (Activation) (None, 32, 6, 6) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_3 (MaxPooling2 (None, 32, 3, 3) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 32, 3, 3) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_1 (Flatten) (None, 288) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 128) 36992

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_2 (Dropout) (None, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 27) 3483

=================================================================

Total params: 133,179

Trainable params: 133,179

Non-trainable params: 0

Fig: Convolutional Neural Network Model