Preprocess the Handwritten Document Image for Preparing Writer Recognition

(A dissertation submitted in partial fulfillment of the requirements of Bachelor of Technology in Computer Science and Engineering of the Maulana Abul Kalam Azad University of Technology, West Bengal)

Submitted by

TAPAN MONDAL (ROLL NO.- 11200116019)
SURAJ AHMED HOSSAIN (ROLL NO.- 11200116020)
SPANDAN MONDAL (ROLL NO.- 11200116025)
RAIHAN AFROZ (ROLL NO.- 11200116038)
ANOWAR HOSSAIN (ROLL NO.- 11200116064)

Under the guidance of
Mrs. Jaya Paul
Asst. Professor
Dept. of Computer Science and Engineering

Government College of Engineering and Leather Technology

(Affiliated to MAKAUT, West Bengal)

Kolakta - 700106, WB

2019 - 2020

Certificate of Approval

This is to certify that the project report on "Preprocess The Handwritten Document Image For Preparing Writer Recognition" is a record of bonafide work, carried out by **Mrs. Prof. Jaya Paul** and **Shri Dr. Santanu Halder** under my guidance and supervision.

In my opinion, the report in its present form is in conformity as specified by Government College of Engineering and Leather Technology and as per regulations of the Maulana Abul Kalam Azad University of Technology, West Bengal. To the best of my knowledge the results presented here are original in nature and worthy of incorporation in the project report for the B.Tech. Program in Computer Science and Engineering.

Signature of
Supervisor/guide

Signature of
Head, Dept. of CSE

ACKNOWLEDGEMENT

With great pleasure, I would like to express my profound gratitude and indebtedness to **Prof. Jaya Paul**, Department of Computer Science and Engineering, Government College of Engineering and Leather Technology, W.B. for her continuous guidance, valuable advice and constant encouragement throughout the project work. Her valuable and constructive suggestions at many difficult situations are immensely acknowledged. I am in short of words to express her contribution to this thesis through criticism, suggestions and discussions

.

I would like to take this opportunity to thank **Prof. Jaya Paul**, Project Coordinator and **Dr. Santanu Halder**, HOD, Department of Computer Science & Engineering, Government College of Engineering and Leather Technology.

I would like to express my gratitude to **Prof. D. Ganguly** and **Prof. S.Sarkar** for their valuable suggestions and help.

Signature

1.Tapan Mondal – 11200116019
2.Suraj Ahmed Hossain – 11200116020
3.Spandan Mondal – 11200116025
4.Raihan Afroz – 11200116038
5.Anowar Hossain – 11200116064

CONTENTS

ABSTRACT

Image preprocessing techniques are the first step for writer recognition system.This paper deals with the various preprocessing techniques involved in writer recognition systems with different kinds of images ranging from simple handwritten form based documents and documents containing colored and complex background and varied intensities. Here, we are going to discuss all important preprocessing techniques like pixel level image preprocessing to derive binary image, histogram adjustment & Threshold to binary image, Filtering, noise removal techniques, morphological processing techniques.

1. INTRODUCTION

Document image analysis(DIA) is a promising area of research. Since two decades, it has attracted a lot of researchers to provide solutions to automation of document processing. Document image analysis mainly deals with recognition and extraction of textual as well as graphical components. Text to speech, signature verification, script identification, detection of name, address, pin-code etc., are the applications of document image analysis. This proposed solution is to detect handwritten areas of a document image and accruing it's pixel data and dimensions so that the handwritten text can be automated to further process for conversion into character strings as well as detection of the author.

1.1 Motivation

Writer recognition has been one of the most fascinating and challenging research areas in the field of image processing and pattern recognition in recent years. Another big benefit is to apply for big documents to convert them for digital storage of historical documents. The acquired image data needs an automated processing (Document Image Analysis (DIA)) and additionally in the case of historical documents a digital restoration. The research topics of this thesis comprise DIA preprocessing, specifically document binarization, document skew estimation and form classification. For writer recognition we have to preprocess the handwritten document image. This project will help us to identify a writer from a handwritten image. This project has made the following contributions in the field of handwritten Bengali, English and Hindi text image preprocess for writer recognition.

- The proposed algorithms of segmentation can be modified further to improve accuracy of segmentation.
- New features can be added to improve the accuracy to detect the handwritten text.
- Recognition of digits in the text, half characters and compound characters can be done to improve the word recognition rate.

1.2 Background

To preserve cultural heritage and human knowledge in the form of written texts and printed books, libraries, national archives and projects like Google Books of Google Inc. started a mass digitization. Additional projects, like Improving Access to Text (IMPACT) and manuscript research centers (e.g. Vestigia - The Manuscript Research Centre of Graz University), have the aim to digitize and improve the access to historical documents. Farahmand et al. [2] discuss different types document image noise and provide review of noise removal methods. Wang et al. presented a fast text detection algorithm for scanned document images. They used low pass filters in their technique to remove high frequency noise, morphological operations and edge detection schemes. Then to enhance the text pixels a clipping operation is applied on L and C channels of LCH space. Khan and Puri presented a study of text detection techniques for printed documents. Peng et al. delivered a method to estimate the quality of documents using sparse representation. [presents the use of sparse representation for restoration of noisy document images. Lelore and Bouchara proposed an algorithm for document image restoration using double thresholded edge detection method. Banerjee et al. employed a probabilistic context model for restoration of document images. They integrated restoration and super-resolution into a single framework. Sgarbi et al.used morphological color operators for color text restoration. A genetic algorithm for document image restoration is developed and presented in Yagoubi et al. presented a nonlinear anisotropic diffusion approach for document image enhancement. This method helped to better distinguish the foreground and background of text characters. A super-resolution technique for text images is proposed. The teager filter is used to obtain super-resolution text. Document image enhancement for Farsi text documents using logarithm domain is presented in . Tan and Shen [proposed wavelet based technique for restoration of archived documents. They developed a method for recovering contents of handwritten documents by interfering

handwriting on the reverse side caused by seeping of ink. Balamurugan et al. proposed steerable filters based on fuzzy unsharp masking. This technique introduces fuzzy based theory for unsharp masking. Image enhancement techniques for printed bangle documents are experimented by Islam et al. . Masood et a detailed survey of features that have been used for content based image retrieval. Nagabhushana et al. present a method for image reconstruction using wavelets. They propose a new multi-level wavelet decomposition for better reconstruction of the images. Thus many techniques and methods for document image enhancement and restoration are reported in the literature.
This project was proposed by Our Project Coordinator to us, and we found this project very interesting as this opened.

1.3 Hardware/Software used

The project was mainly developed in 2 dell inspiron 3558 laptops with intel core i5 processor /4 gb DDR3 ram. The testing was done with 2 more laptops. One with intel core i3 / 4gb ram(dell inspiron), another with intel core i5/ 8gb DDR4 ram(hp pavilion).

Windows 10 64bit was our main choice for operating-system. However the project was also tested in Ubuntu 15.04 LTS (One of the machines had Dual boot).

We basically used visual studio code (VSCODE) for our editor of choice.In somecases we used SUBLIME TEXT as an editor. For Interpretation we used Python3.7 64-bit interpreter. For package management pip was used.

For version control and collaboration git was used and as online cloud backup we used github.com

2. DATASET

2.1 Provided Dataset

The Provided dataset consists of 101 writers and almost each writer has 5 sets of Bengali, English and Hindi language data set files and the total number of data files is 1360. The datasets are provided in 6 folders named as Constrain1(184 data files), Constrain2(206 data files), Constrain3(69 data files), Constrain4(110 data files), Constrain5(439 data files) and Constrain6(210 data files).

The data are provided as an image file and the dimension is 2481 x 3507(pixels).

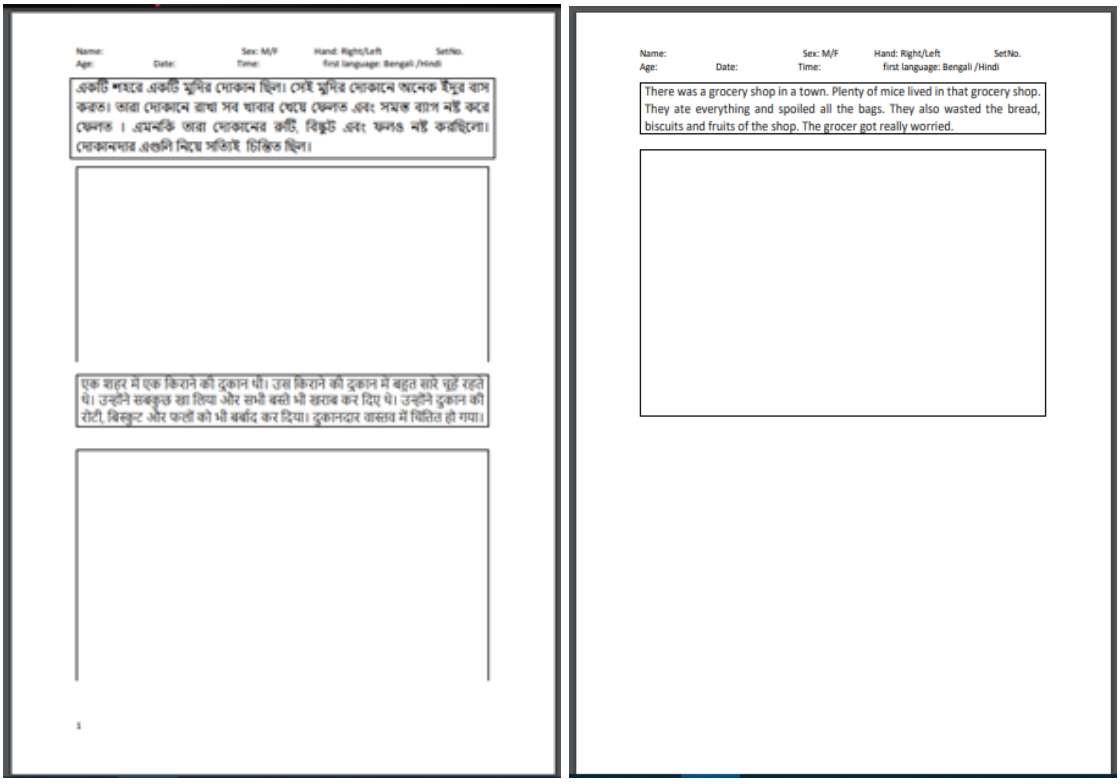


Fig. 2.1: The left form is for bengali & hindi data and the right form is for english data

Fig. 2.1 shows the format in which data is collected. Data is collected from people with different ages(18 to 60). All the writers are affluent in english. Those who know bengali filled the above dataset form in bengali and those who know hindi filled the dataset form in hindi. Balck and blue ink ballpoint pen is used to fill the form. A small paragraph is used in three different languages to collect the data. Various parameters like writer's name, age, gender, set number, date and time of writing, first language and hand of writing are collected for further analysis.

2.2 Derived Dataset

We run a python script through the provided data set and slice the data and we get around 20000 sample data from the sliced data. Then by giving some data to the training model randomly from our sample data and by applying shear, rotation and scaling we get more data. We separate the potential training and validating data and keep them in two folders(dataTrain and dataValidation). Each folder has two subfolders naming hand and other. We manually separate the sliced hand written and non handwritten data.

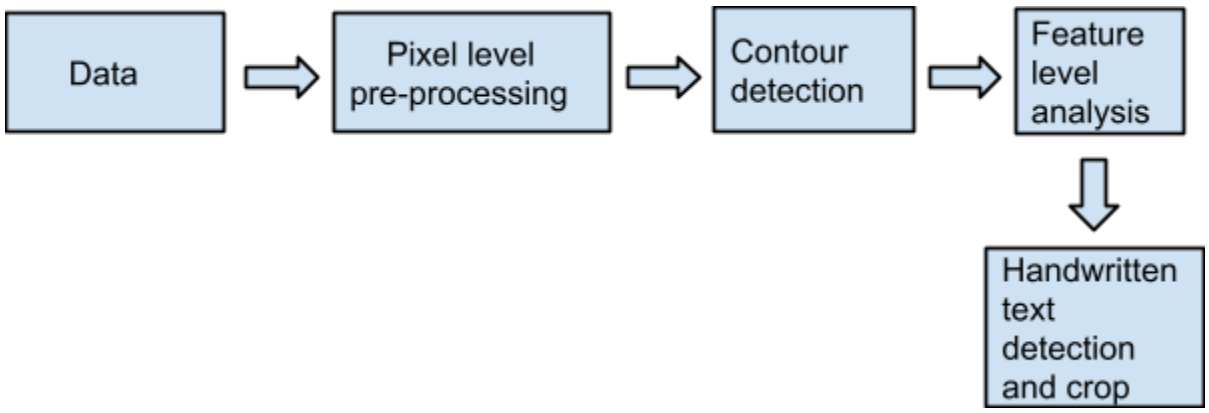
We also use a subset of data for testing purposes. Some of which are common with training data. Some of which are unique, exclusively for testing. .

After completing paragraph extraction we separated the paragraphs of each language separately i.e. Bengali, English and Hindi. These served as a testing dataset for our word separation algorithm.

3. METHODOLOGY

The image samples that we are working with belong to raster images. These images are typically represented as a 2-dimensional array of pixels, where each pixel has 3 values for red, green and blue channels. Each channel is represented using an unsigned byte thus can have value between 0 to 255. The raw image data must be derived in the mentioned formal before further processing.

The main purpose is to detect and organize the handwritten portion of the document image for further analysis. Thus we have tried to first detect handwritten text blocks as paragraphs from individual images. From those paragraphs we further tried to separate all the words as cropped images. The whole process is divided into multiple stages creating a linear pipeline to follow -



3.1 Pixel-level Image Preprocessing to derive binary image

Pixel level processing includes many successive steps one after another. However the order must be kept to avoid unexpected results. The main purpose of pixel level processing is to deform the image in order to make it simple for the computer to calculate contour[1]. First of all all unwanted objects such as noise, dark spots and lines must be detected and removed in order to have only text data and whitespace in the image. After that the primary color ranges for the texts inside the image must be determined and all pixels that have a near value to these ranges must be darkened and sharped. This makes detection of text easy from the document image. After this the values of pixels must be normalized between black(255) and white(0) two individual binary levels representing presence and absence of information.

3.1.1 Noise removal

Noise can be represented as small dots or paper grains as well as large shadows that might be casted onto the image. So we go for two stages -

Large noise removal: To remove any shadow spots or large noise first we separate the red green and blue channels(plains) as individual matrices of dimension width x height .Then we dilate the single channel matrices and store the value. Then create a highly blurred version from the dilated image so that no text is recognisable and store the value. This is typically done by applying a gaussian blur. Finally subtract the data of the blurred channel matrix from the original channel matrix to keep only the necessary information. Then for each plain we repeat these processes.Marge all the channel matrices together to form the rgb image with large noise removed.

Small noise removal: A low pass filter is applied to dilate the image to remove small noises[3]. With the same lowpass filter erosion[3] is applied on the image to make the text portion sharper.

As we see in Fig. 3.1 the left side document image was full of shadow and Small Noises however in the right-side all the noises and shadow spots were removed successfully.

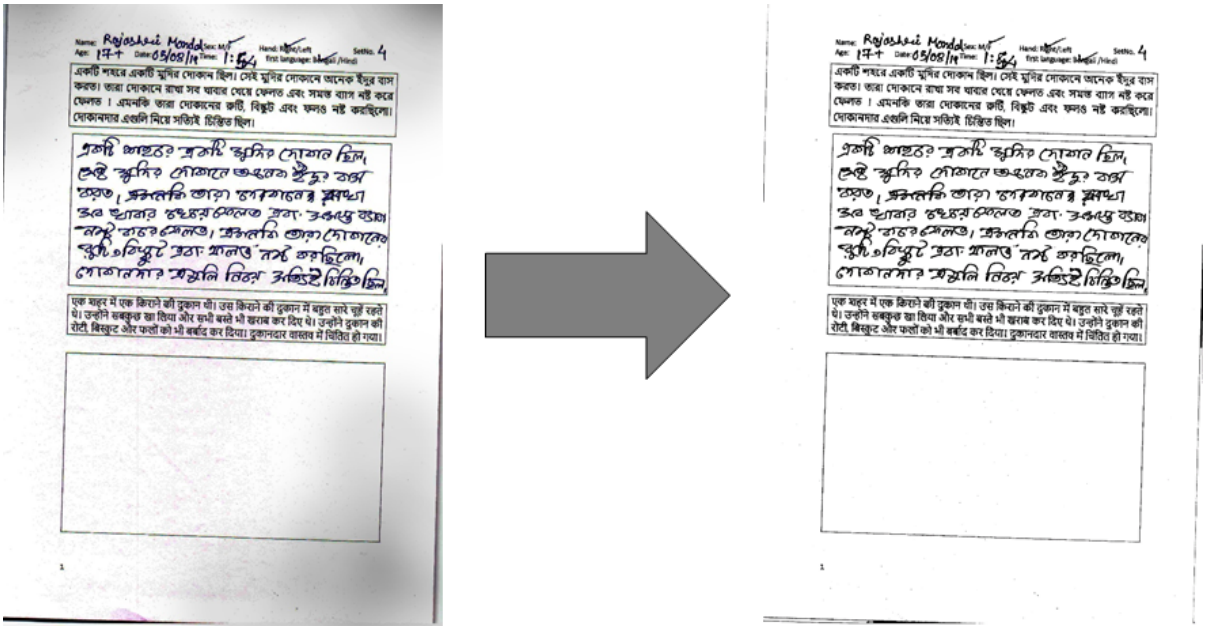


Fig. 3.1: Noise removal form document image

3.1.2 Line detection & removal

In our test samples the document images contained boxes around the text. So it was necessary to remove these lines for further preprocessing. Lines can be represented as [y = mx + c], where(x, y) are the coordinates on the line. m is the tangent of the angle i.e. dyd/x and c is the value of y where x = 0. This can easily be converted into polar coordinates i.e. (θ, r) where y = r*sinθ and x = r*cosθ. A process to detect lines in this format is known as hough transform[4] and can easily be done on pixel arrays to determine lines. So to remove the lines from the cleared image -

At first we take an otsu[6] threshold copy of the image. To get that we copy the cleared image, convert it into gray scale and apply threshold.Then we detect lines using probabilistic hough transform(PHS). Nearby segments with a minimum gap are considered as the part of the same line. Now the only lines that are longer than a minimum length are considered as positive detected lines.After detecte those lines these are overlaid with white pixels(255,255,255) into the cleared image to obtain the line free clear image.

Left image in Fig 3.2 shows detected lines in green color and at right the lines are removed by overlapping white color over the lines.

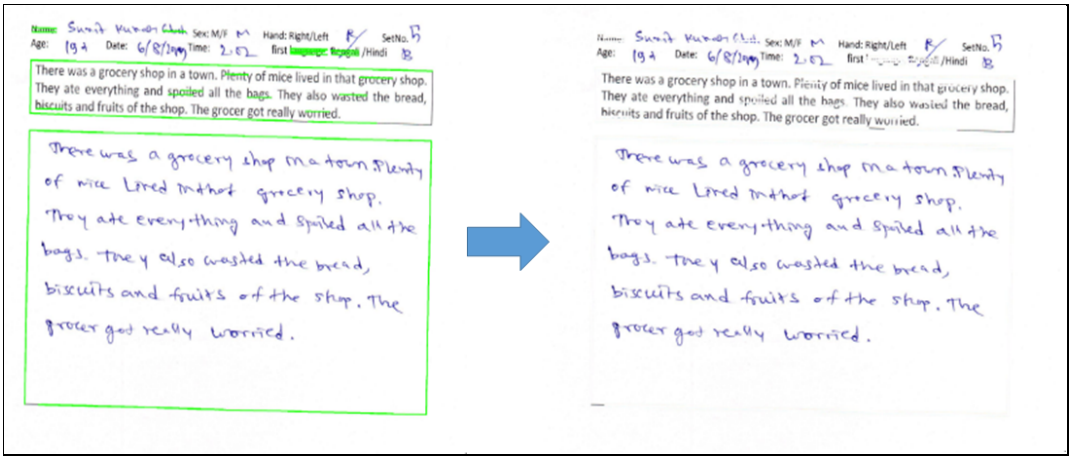


Fig. 3.2: line detection and removal

3.1.3 Histogram Adjustment & Threshold to binary image

The text part should be highlighted and the rest part should be completely ignored so that further processing is possible[5]. To Achieve this we do as following-

First we convert the derived image into a grayscale image. As the image is in grayscale at this stage it contains only one channel that can have value between 0 to 255. Then we take a histogram of the grayscale image.Then We decide a small bias and find the picks from the histogram which are local minima. Figure 3.3 shows the histogram of a grayscale document image. The pick between 150 to 200 describes white color density. We can find the minima in between 0 to 50 in the figure. These represent the colors of the text. After that the values are converted into ranges of [value - bias] to [value + bias]. Then the Pixels that belong to any of the range are replaced with black pixels (0). All other pixels are replaced with white pixels (255). Finally once again otsu threshold[6] is applied to the final image and this image is stored as the main reference to be used for contour detection feature level analysis.

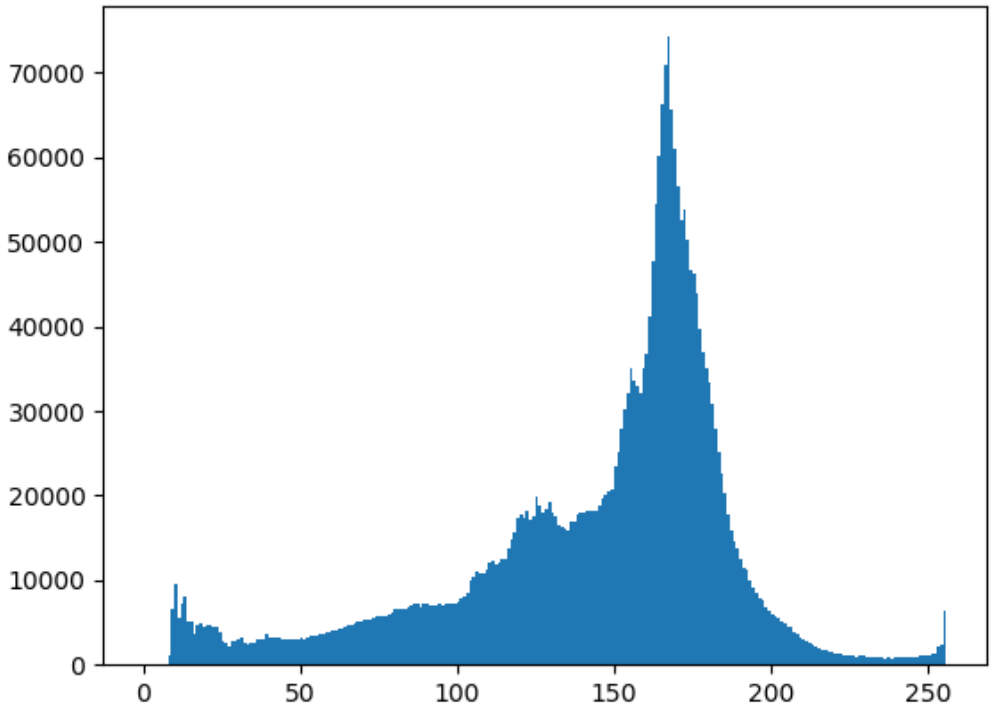


Fig. 3.3 : Histogram of a document image

3.2 Morphological transform & Contour Detection

Morphological Transforms are used as filters to find different general structures and shapes of objects inside the image. Filters are basically two dimensional arrays i.e. matrix sometimes also known as a kernel. These are multiplied with the image matrix to produce deformed images. Here we have used three morphological transforms for different kinds of contours. The general procedure is to -

At first we apply the specific morphological transform according to need. Then detect all the contours as rectangles from the transformed image. In most of the cases we try to keep the aspect ratio (width /height) of these rectangles greater than 1.3 as most paragraphs and words are wider than their height. If this condition breaches we once again apply the same procedure with a secondary filter with low resolution onto the specific contour in order to further analysis as in most cases this denotes that two unwanted components are connected. For our case we retrieve the external contours using simple chain approximation to get better results. These results are generally retrieved as an array and may contain both handwritten as well as printed text. Further classification is only possible through a trained machine learning model to determine whether the text is handwritten or printed.

3.2.1 Rectangular filter for paragraph and line

To detect paragraphs and lines we apply rectangular structural filters for morphological transform[2].Our rectangular filters are mainly horizontal rectangles that roughly match the shape of a paragraph and line. For paragraphs we used a dimension of (18: 6) for lines we use a dimension of (100: 5). Fig 3.4 shows the output of application of rectangular morphological shape transform upon the threshold of the left image at the right. As we see all the components that belong to the same paragraph get connected together to denote a single component. Paragraph contours that have aspect ratio over 2 are sliced horizontally into 2 equal pieces in order to preserve information for the machine learning model. When these contours are fed to the model they are resized to a fixed size. So doing so helps a lot in increasing the accuracy of the machine learning model.

Here is a demo filter for paragraph detection -

```
[[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]]
```

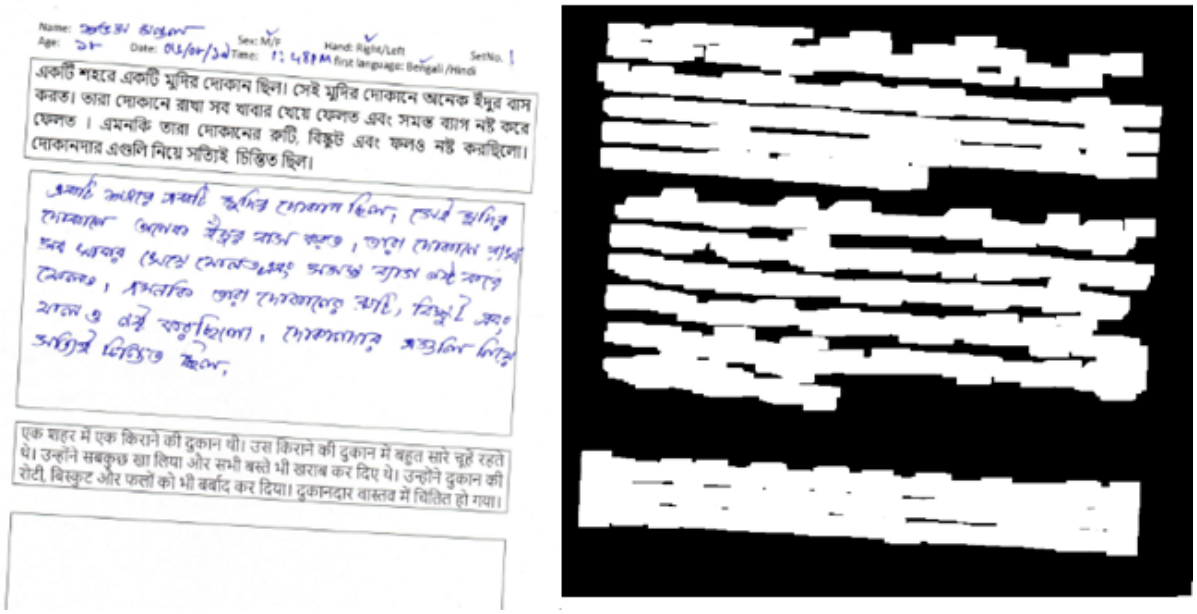


Fig. 3.4: Threshold image after rectangular morphological transform

3.2.2 Anisotropic filter for words

For printed texts small morphological rectangles are good filters but in mostcase they fail in case of detecting handwritten words. For this case we found anisotropic filters to work very well. These filters are uneven and help to join small components of the words more effectively. Fig 3.5 shows an example cleanup to binary image(middle) and then applying anisotropic filter[2] to the threshold image.

Here is an algorithm to produce an Anisotropic filter -

First of all we take three parameters kernel size (K), sigma (S) and theta (T). we derive half size(H) where H = $\sqrt{K}$. Next we create a matrix(Kernel) of size[K x K]. We derive 2 new constants as Sx and Sy where Sx = S and Sy = S * t . Now we loop through form 0 to K for column and 0 to K for row and derive x = row index(i) - H and y = Column index(j) - H. Now we calculate the exponential term (E), x term (Ex) and y term(Ey) based on the below mentioned formulae -

$$E = (-x^2/(2 * Sx) - y^2/(2 * Sy))$$
$$Ex = (x^2 - Sx^2)/(2 * \Pi * Sx^5 * Sy)$$
$$Ey = (y^2 - Sy^2)/(2 * \Pi * Sy^5 * Sx)$$

Finally we set $Kernel_{i,j} = (Ex + Ey) * E$ and perform a matrix division by the sum of all the terms of the Kernel matrix and return the kernel matrix.

Here is a demo anisotropic filter -

[[0.01181348, 0.01211876, 0.0123057, 0.01236866, 0.0123057, 0.01211876, 0.01181348]
[0.01898174, 0.01947225, 0.01977263, 0.01987378, 0.01977263, 0.0194722, 0.01898174]
[0.02507714, 0.02572516, 0.02612199, 0.02625562, 0.02612199, 0.02572516, 0.02507714]
[0.02749211, 0.02820253, 0.02863757, 0.02878407, 0.02863757, 0.02820253, 0.02749211]
[0.02507714, 0.02572516, 0.02612199, 0.02625562, 0.02612199, 0.02572516, 0.02507714]
[0.01898174, 0.01947225, 0.01977263, 0.01987378, 0.01977263, 0.01947225, 0.01898174]
[0.01181348, 0.01211876, 0.0123057, 0.01236866, 0.0123057, 0.01211876, 0.01181348]]

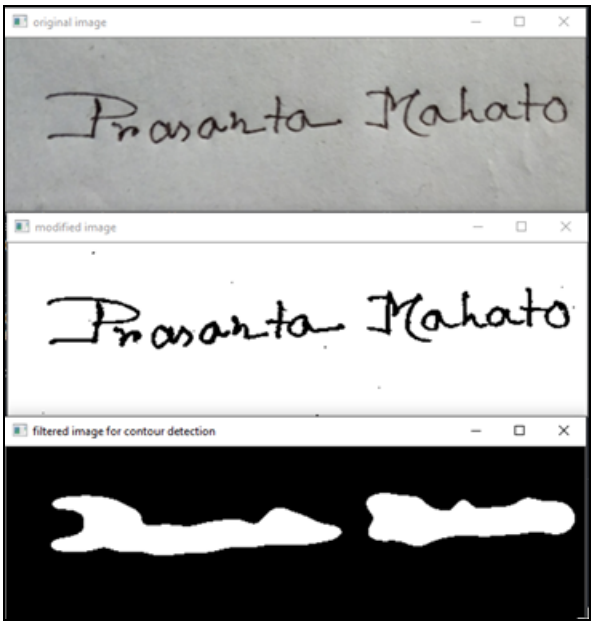


Fig. 3.5: Application of anisotropic kernel as filter

3.3 Feature Detection using convolutional Neural network

At this point we had blocks that contained text but we needed to determine whether the text blocks were printed or handwritten. This required a generic solution. For this we used convolutional neural networks to give us the prediction weather a contour was handwritten or printed. This was a machine learning model that we trained using the dataset that was provided to us. As convolutional models work in 2d space[7] we found it to be a good match for the generic model we seeked to extract features from the contour images. After the convolutional layers extracted their features, we passed the features to a recurrent neural network to process and determine if a piece of text was handwritten. We roughly worked with 64 features. With enough sample training data we actually managed to get a decent accuracy.

3.3.1 Architecture

Neural networks are basically formed by layers after layers of neurons in different arrangements. Here is the generic structure of our implemented network -
Layer 1:
Layer type : convolutional layer(large)
Activation function : Rectified Linear Unit (ReLU)

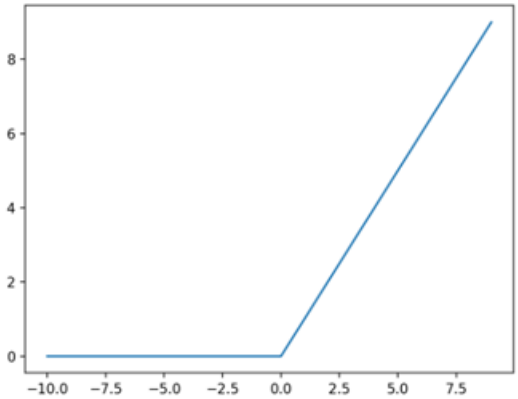


Fig. 3.6: Rectified Linear Unit (ReLU)

Layer2:
Layer type : convolutional layer(medium)
Activation function : Rectified Linear Unit (ReLU)

Layer3:
Layer type : convolutional layer(small)
Activation function : Rectified Linear Unit (ReLU)

Layer4:
Layer type : Recurrent layer
Activation function : Rectified Linear Unit (ReLU)

Layer5:
Layer type : output layer
0 : Non-handwritten
1 : Handwritten
Activation function : sigmoid

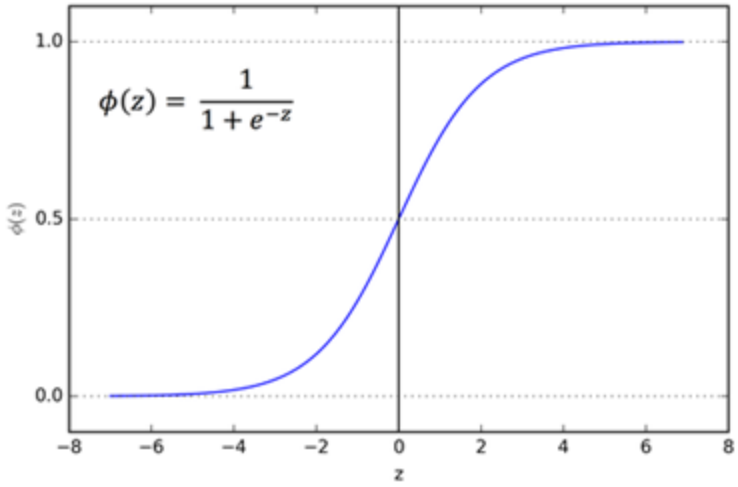


Fig. 3.7: Sigmoid Activation

We introduced dropouts in the middle of the layers to increase the efficiency of the network. Fig. 3.6 and fig 3.7 shows the x vs y plotting for relu and sigmoid activation functions. We also used max pooling in between the convolutional stages in order to reduce unnecessary information.

3.3.2 Training & Testing

The model was trained using over 20000 samples of small image crops generated from the sample images provided to us. For this phase of training we generated the crops using scripts and separated them into two distinct categories (i.e. handwritten and others) manually. After that the data was organized into training and validation sets. While training we randomly modified the images to produce separate data each time by skewing and rotating via a small margin.
As the loss function for training we found binary_crossentropy to be very effective and for optimization the best results were given by rmsprop optimizer. After training for multiple epochs (around 35 we got the best result) it was quite efficient to distinguish between handwritten and others with good accuracy. Fig 3.8 shows the change of the validation accuracy with increase to epochs while training the model.

All samples must be converted into binary format[5] between [0 -1] before feeding as well as are to be resized to the given input size first. After some manual testing we found that 0.5 was a good value for the threshold probability between handwritten texts and others. We tried to have a good number of images different from training set in the testing set. It helped us a lot inorder to properly understand the real progress of our machine learning model.

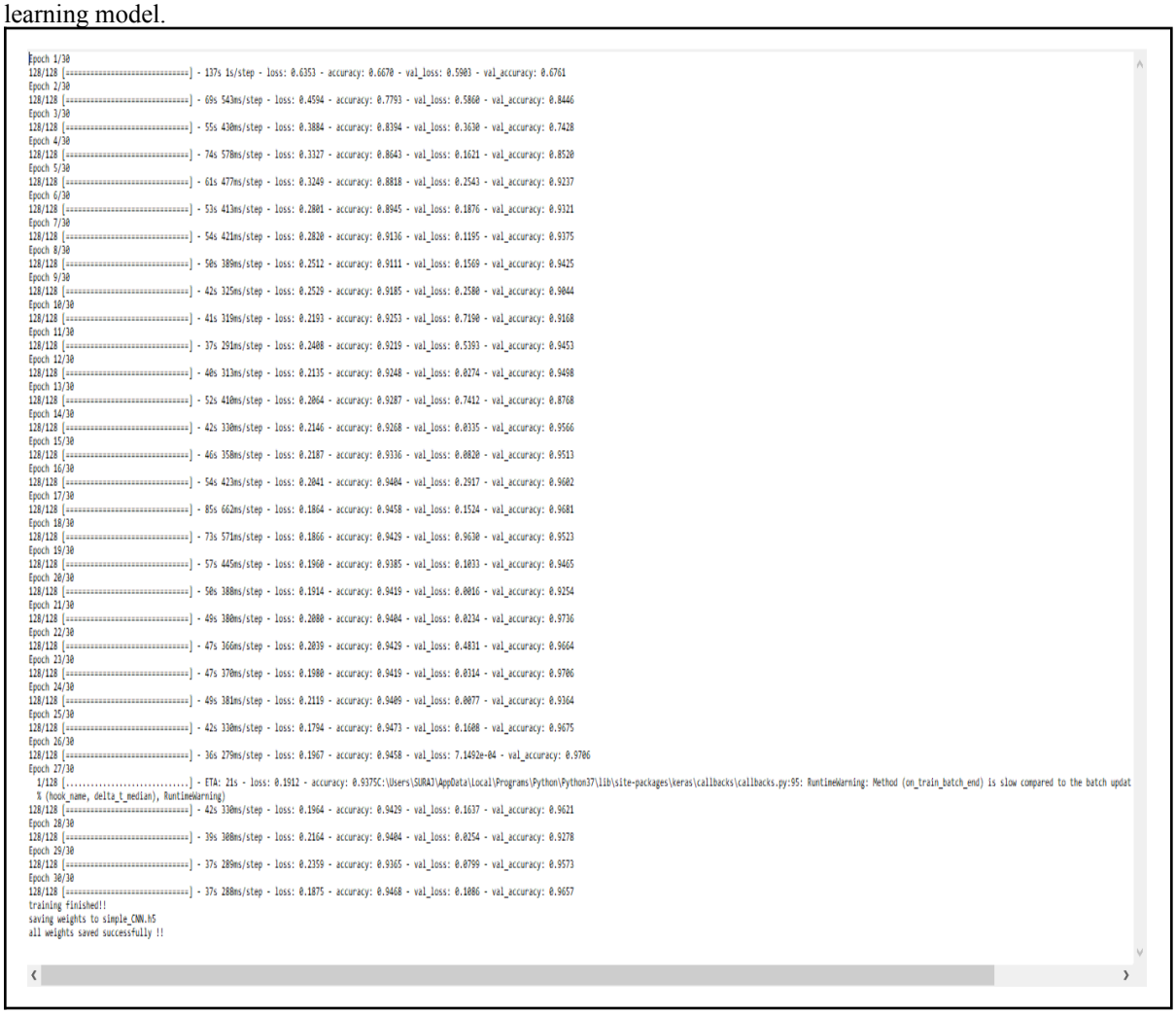


Fig. 3.8: Value of loss and accuracy during training

3.4 Handwritten Paragraph Detection

Till now we had a way to find text regions and a way to validate the data if it was handwritten or not. Now we only needed to combine these into an algorithm to derive our final result as the dimensions and the cropped image of the handwritten paragraph. These paragraph regions are very important for deriving handwritten words. The algorithm is as follows:

For this at first we need to clear the image and remove lines from the image. After that detect contours. Then predict if the contours are handwritten via our machine learning model and divide them into two sets - positive and negative. After the division is complete we use these two sets to find the cluster of handwritten texts and calculate the minimum bounding rectangle. Then we crop the part of the original image which is inside the bounding rectangle that is the handwritten part. After cropping is over we trim the crop part to remove excess white space.
Fig 3.9 shows our detected bounding rectangle that is covering the handwritten text on the line removed document image in green color. This region can easily be cropped to get the paragraph.

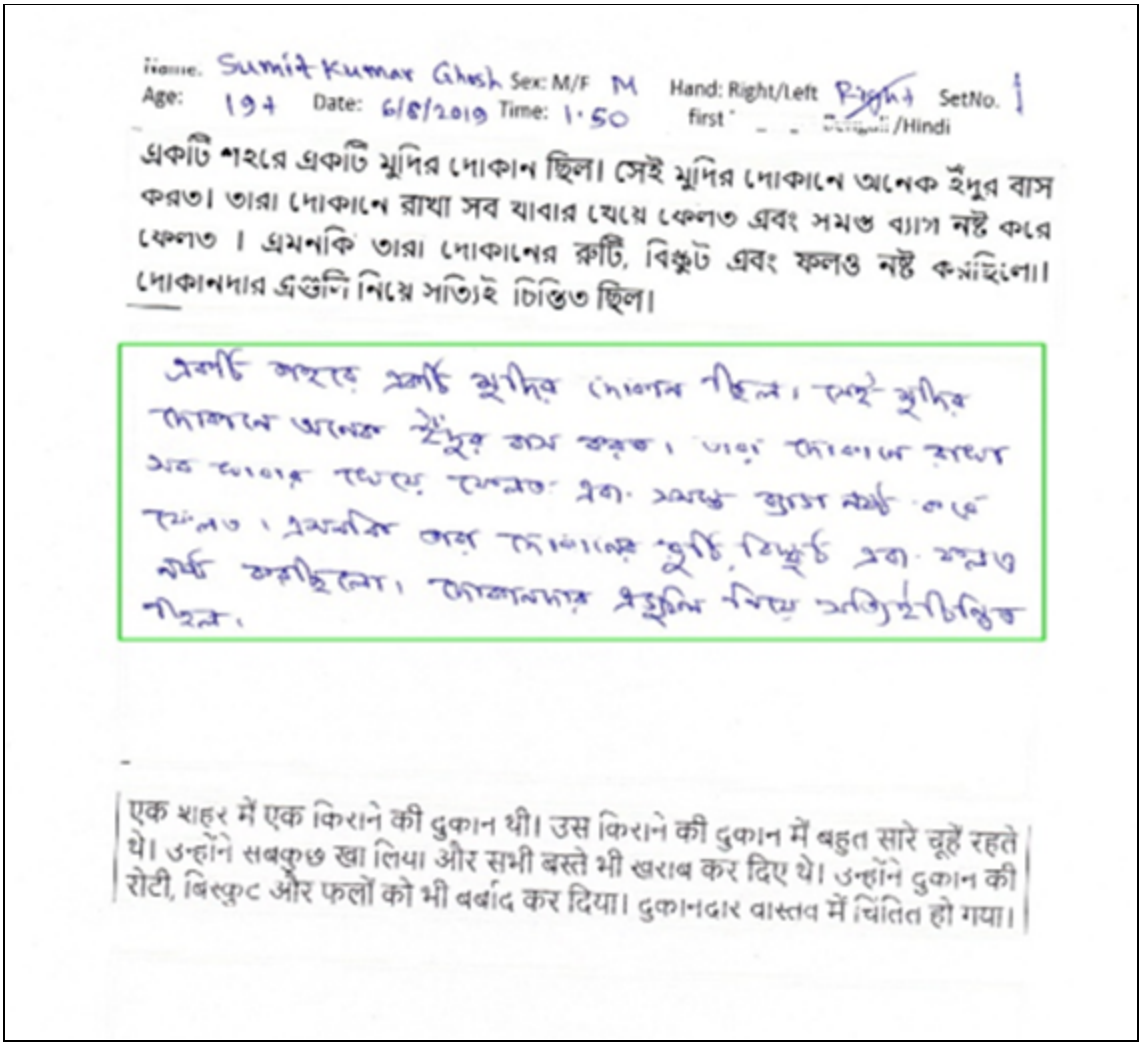


Fig. 3.9: Handwritten paragraph detection

3.4.1 Contour Prediction

For contour prediction at first the contour binary crop images are normalised form[0 - 255] integral range to[0-1] fractional range. These are then resized to fit in the machine learning model. After that two arrays are taken to store positive and negative results. Then the contour prediction runs over the crop images , If the prediction is handwritten the crop is pushed to positive results array, else it is pushed into negative results array.

Fig. 3.10 shows the detected paragraph contours after being predicted by the machine learning model. The contours that are marked in red are thought as negative results by our prediction model and the contours marked in sky color are predicted as positive results for handwritten text. Contours that are too small in the size are removed before running the algorithm.

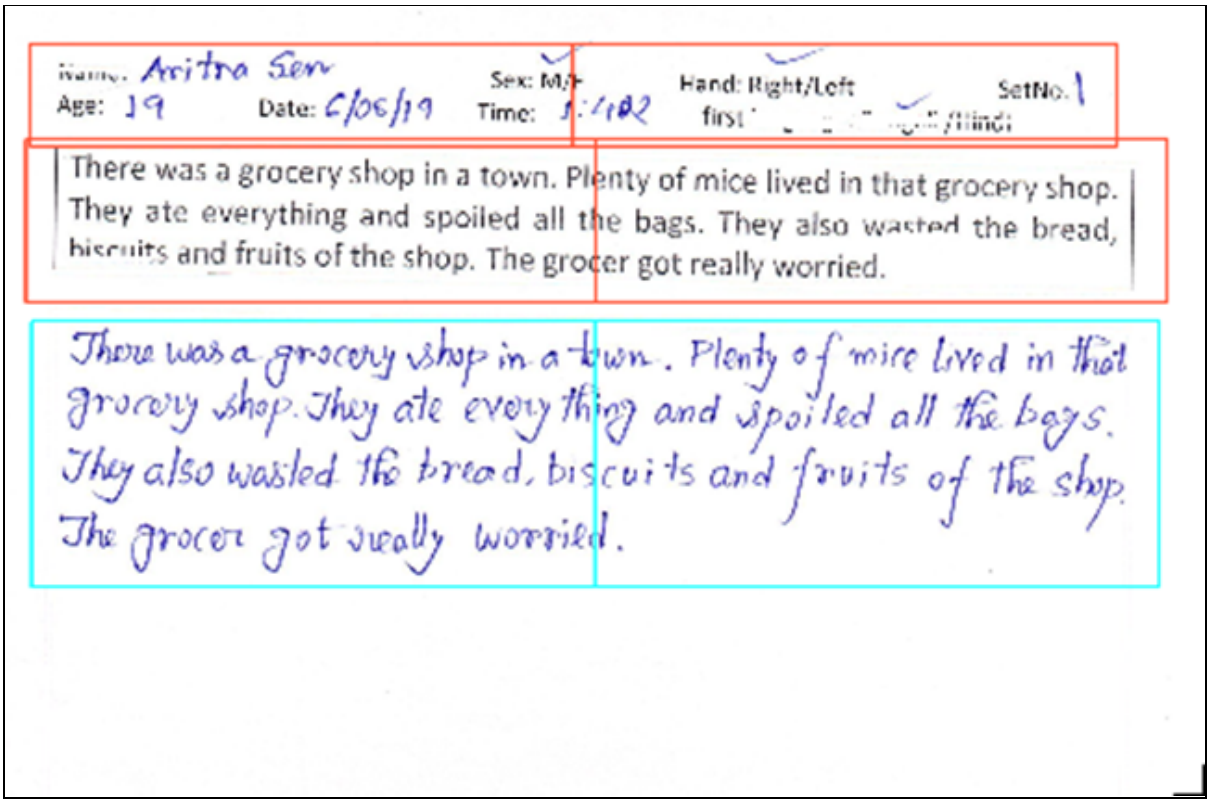


Fig. 3.10: Contour prediction(RED: negative, SKY: positive)

3.4.2 Contour Cluster arrangement for minimum rectangle

Here we use a function to determine the minimum bounding box for the handwritten paragraphs.It iterates through the positive and negative list and finds the nearby paragraphs finally computing the bounding rect. To eliminate minor negative results we use a minimum area fraction, its value is 0.01 for our case. The algorithm for the function goes like this -

First we create an array to store all the paragraphs. Each paragraph is an array itself that holds multiple contours which are x, y, width(w) and height(h). Initially our algorithm takes 2 inputs as positive array of contours and negative array of contours. We define a minimum area which is equal to the image width * image height * minimum area fraction(0.01 in out case). We also define a minimum space called Wspace which is equal to image width * space factor(we take 0.1 as default value). We also take two markers as currentHeight and PreviousHeight to keep track of our virtual cursor over the page. We create another additional array of contours to temporarily hold our potential candidates called current segments. Now we loop through all positive contours and check if the absolute value of(y - currentHeight) > wspace and (y - previousHeight) > Wspace if so we push them into the current segments array. And check if any negative conture is in the vicinity. If the percentage of negative samples is more than 20% we discard the current segment else push the current segment into the paragraphs array. Finally we calculate the minimum bounding rect for all the paragraphs and return them.

3.5 Handwritten word Detection

After the paragraphs are cropped they serve as the base image for word detection. Firstly we apply Anisotropic Morphological transform on the paragraph image and again do contour detection. This time we loop through each image and if any vertical rectangular contour is found we dilute the contour image and again run through the contour detection algorithm to reduce chances of vertically connected words.

After the word contours are found we finally sort them vertically using their y coordinate in the page to make them arranged and return to the word cropping algorithm.

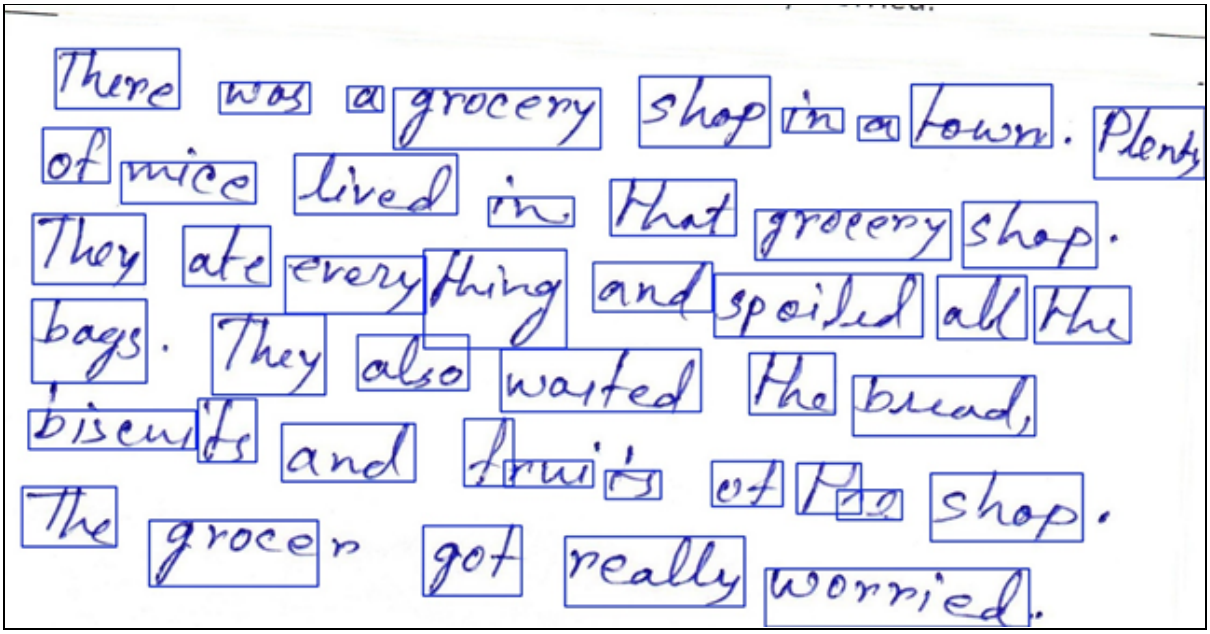


Fig.3.12: Word contour detection for word cropping

4. Architecture

Our whole architecture pipeline can be divided into many stages. These stages involve image preprocessing as well as application of a machine learning model inorder to maximize the accuracy of the project. Fig. 4.1 demonstrates our general architecture in all distinct stages. The purple box i.e. primary image-preprocessing is a combination of multiple stages, thus it is discussed separately. Our architectecure was implemented using python 3. We used opencv as our primary computer vision library to import and manipulate image data. We used keras from tensorflow to implement our machine learning model and tkinter for gui(Graphics user interface) implementation. Our program produces primarily handwritten paragraphs from a document image and uses these paragraphs to further produce individual handwritten words. We first preprocess the image to enhance it and pass to contour detection algorithm after thresholding. Different filters are used for different kinds of structural component detection. These are mentioned in table 4.1 . These components i.e. contours are passed over crop detection algorithms as mentioned in fig 4.1 to detect paragraph and word contours. In these steps we always use a min area fraction of 0.001 to ensure that any subcomponent is not selected as an independent component.

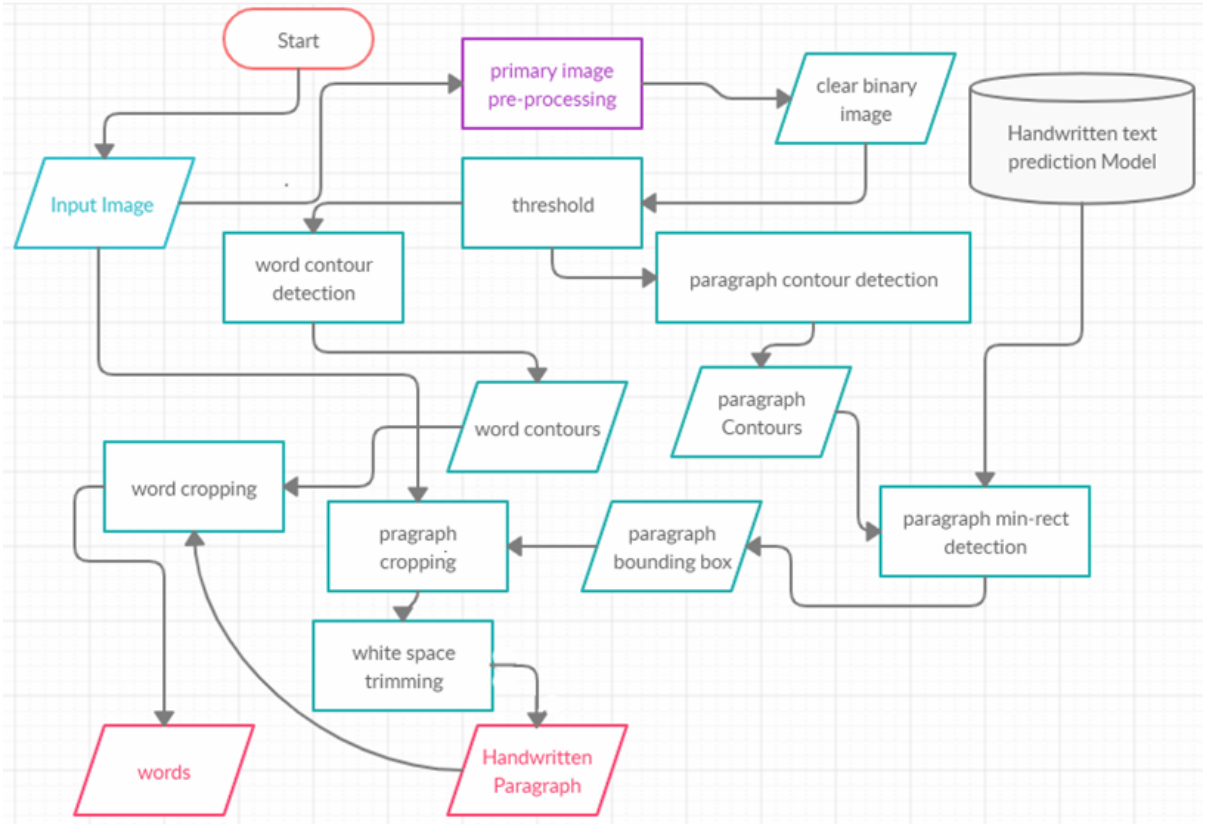


Fig. 4.1 : General architecture of whole pipeline

Table 4.1 - Filter kernel values

Contour type	Filter kernel type	Filter params
Paragraph	Rectangular structural element	Kernel size = 7 x 7

		Kernel cell value = 1
words	Anisotropic	Kernel size = 37 x 37 Sigma = 15 Theta = 14
both	Gaussian blur	Kernel size = 5 x 5
Erode and dilute	Rectangular structural element	Kernel size = 1 x 1

4.1 Image Preprocessing

The steps involving preprocessing of the image for contour detection are shown in fig 4.2 and the critical parameters are mentioned in table 4.2. The in depth descriptions for all methods can be found in the methodology section. We have tried different values for different parameters and these values gave us optimal results.

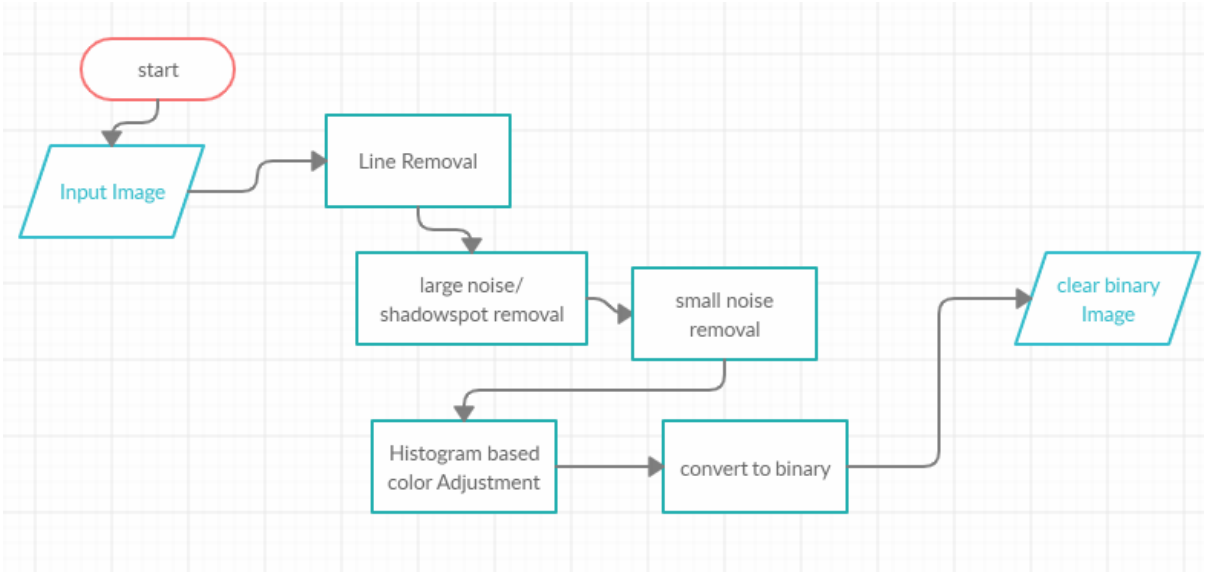


Fig 4.2: Image pre-processing pipeline

Table 4.2: Critical values for parameters

Stage	Method	Critical values
Line Removal	Probabilistic Hough transform	Minimum line length=100 Maximum line gap=10 Delta angle = 1deg
Shadow spot removal	blur	Type = median Kernel size = 21 x 21
Shadow spot removal	normalization	Type = minmax Min = 0 Max = 255

Color Adjustment	Histogram shift	Minimum value bias = 40 Maximum value bias = 70 Foreground value = 255 Background value = 0

4.2 Neural Network

The neural network model uses a convolutional model attached with a recurrent model. The full diagram is provided in fig 4.3 and the parameters are provided in table 4.3. Net accuracy matrices can be found in section 5.2

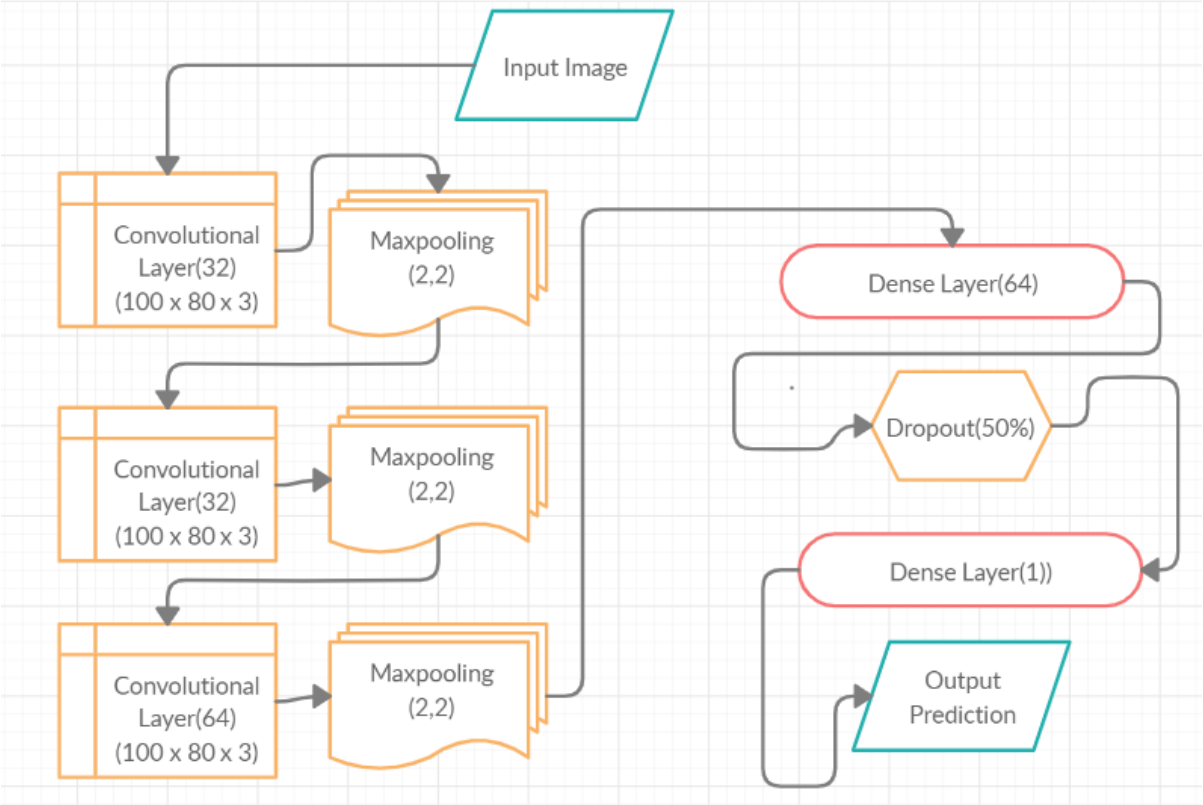


Fig 4.3 : diagram of the model

Table 4.3 - layer description in order

Sl no,	Layer type	size	Additional configuration
1	input	100 x 80 x 3	
2	Conv2D	100 x 80 x 3, 32	Max Pooling = 2x2, activation relu
3	Conv2D	100 x 80 x 3, 32	Max Pooling = 2x2, activation relu
4	Conv2D	100 x 80 x 3, 64	Max Pooling = 2x2, activation relu
5.	Dense	64	Dropout = 50%, activation relu
6	Dense, output	1	Activation sigmoid

4.3 Limitations

In spite of these efforts there are still some limitations of our architecture. As our project deals with non homogeneous data, i.e. there is no fixed absolute rule that governs over the input data, the architecture might fail to perform 100% efficiently. Some of these limitations are discussed here -

- uneven distribution of text and presence of overlapping
- variation of handwriting
- variation of languages (we are mainly testing on Bengali, English and Hindi samples)
- accuracy of cnn model for prediction
- sample data size
- font size and lighting conditions (not so important)

5. EVALUATION MATRICES

We measure the performance on the basis of result. The code is written in python. As python is an interpreted language it is slower than C or C++. The program is using linear time to execute if we take no. of images as the size of input.

5.1 Performance Benchmarking

The Python time module is used to benchmark the performance of our algorithms. Performance is measured in three different machines.
Machine specifications : Intel core i3/4GB RAM, Intel core i5/ 4GB RAM, Intel core i5/ 8GB RAM.

5.1.1 Pixel-level Image Preprocessing to derive binary image

Table 5.1 shows the time taken to perform pixel level processing on a single document image of size 2481 x 3507.

Table 5.1: Pixel-level Image Preprocessing to derive binary image

Machine specification	Time (microseconds)
Intel core i3/4GB RAM	2760 - 3570 approx.
Intel core i5/4GB RAM	2550 - 2800 approx.
Intel core i5/8GB RAM	2530 - 2660 approx.

5.1.2 Morphological transform & Contour Detection

Morphological transformation for total images takes approximately 1500 - 2000 microseconds in our testing environment.

5.1.3 Handwritten Paragraph Detection

The whole process from raw image input to handwritten paragraph detection takes about 1/10th of a second in approx. table 5.2 explains what we found in our testing environments.

Table 5.2: Handwritten Paragraph Detection

Machine specification	Time (microseconds)
Intel core i3/4GB RAM	60000 - 100000 approx.
Intel core i5/4GB RAM	40000 - 60000 approx.
Intel core i5/8GB RAM	30000 - 60000 approx.

5.2 Paragraph extraction matrices

After we ran our algorithm through all the samples that were provided as a final test we counted all the successful outputs. Our algorithm was right to detect through about 94% of the samples provided. The matrices of the results are shown in table 5.3 and 5.4.

Table 5.3: Number of successful outputs for each input

Set	Language	Number of Sample	Number of Successful Output
Constrain1	BENGALI	64	62
	ENGLISH	92	86
	HINDI	81	79
Constrain2	BENGALI	77	74
	ENGLISH	104	99
	HINDI	75	74
Constrain3	BENGALI	35	33
	ENGLISH	34	33
	HINDI	32	32

Constrain4	BENGALI	45	41
	ENGLISH	55	50
	HINDI	10	10
Constrain5	BENGALI	168	157
	ENGLISH	214	207
	HINDI	64	59
Constrain6	BENGALI	105	94
	ENGLISH	105	92
	HINDI	0	0

- Overall accuracy is 94.47%

Table 5.4: Paragraph extraction accuracy of each language

Language	**Accuracy (%)**
Bengali	92.91
English	93.54
Hindi	96.95

5.3 Word extraction matrices

After paragraph extraction word extraction is done. English language word extraction success rate is higher compared to bengali and hindi. Table 5.5 shows the overall accuracy of our word extraction algorithm on cropped paragraphs.

Table 5.5: Word extraction accuracy of each language

Language	**Accuracy (%)**
Bengali	73.29
English	80.23
Hindi	65.75

CONCLUSION

Preprocessing techniques used in document images as an initial step in character recognition as well as writer recognition systems were presented. Future research aims at new applications such as online character recognition used in mobile devices, extraction of text from video images, extraction of information from accuracy security documents and processing of historical documents. The objective of such research is to guarantee the security of information extraction in real time applications. Even though many methods and techniques have been developed for preprocessing there are still problems that are not solved completely and more investigations need to be carried out in order to provide solutions.

Most preprocessing techniques are application-specific and not all preprocessing techniques have to be applied to all applications. Each application may require different preprocessing techniques depending on the different factors that may affect the quality of its images, such as those introduced during the image acquisition stage. Image manipulation/enhancement techniques do not need to be performed on an entire image since not all parts of an image is affected by noise or contrast variations; therefore,enhancement of a portion of the original image may be more useful in many situations. This is obvious when an image contains different objects which may differ in their brightness or darkness from the other parts of the image; thereby, when portions of an image can be selected,either manually or automatically according to their brightness such processing can be used to bring out local detail. In conclusion, preprocessing is considered a crucial stage in most automatic document image analysis systems and without it the success of such systems are not guaranteed.

Automated handwritten character recognition seems to be necessary due to the increasing number of online writer identification applications.

Handwritten document processing has become an inherent part of the office automation process. Automatic identification facilitates the reading and processing the multi-script documents for various applications. The objective of this project is to identify handwritten characters with the use of neural networks. Detecting the text strokes based on histogram features .Strokes can be used to recognize the text using trained characters.Finally the aim is to implement a deep learning approach such as a neural network algorithm to classify the characters with author details.

REFERENCES

[1] IEEE Computer Society Executive Briefings Document Image Analysis Lawrence O'Gorman Rangachar Kasturi ISBN 0-8186-7802-X Library of Congress Number 97-17283, 1997

[2] Umesh D. Dixit and M.S.Shirdhonkar, "A Survey on Document Image Analysis and Retrieval System", International Journal on Cybernetics & Informatics, 4(2), 2015, 259-270.

[3] Hossein Nezamabadi-pour and Saeid Saryazdi, "An Efficient Method for Document Image Enhancement", Proc. of International Symposium on Telecommunications, 2005, 175-180.

[4]] H. Y. Chen, Y. Y. Lin, and B. Y. Chen, "Robust Feature Matching with Alternate Hough and Inverted Hough Transforms", in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 2762 - 2769.

[5] I. Pratikakis, K. Zagoris, G. Barlas, and B. Gatos, "Icdar2017 com-petition on document image binarization (dibco 2017)," in Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on, vol. 1, pp. 1395–1403, IEEE, 2017.

[6] N. Otsu, "A threshold selection method from gray-level histograms," IEEE transactions on systems, man, and cybernetics, vol. 9, no. 1,pp. 62–66, 1979.

[7] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3431–3440, 2015.