CS574: COMPUTER VISION USING MACHINE LEARNING

Group 22: Handwritten Document Image Analysis

Monika Khandelwal	184101026
Shavi Gupta	184101033
Shubham Jain	184101035
Tushar Geetey	184101039

Abstract

The aim of the project is to convert the handwritten document into digital form. We have used CNN with various architecture along with RNN to classify character within the document. The character are then reconstructed to form a word. We have used Best Path decoding method and wordbeam search method to get the final output.

Introduction

There are thousands of manuscripts, papers that are handwritten but are of no use as they are not digitalized or digitally available. This makes searching through them impossible. Our aim is to digitize the handwritten document.

Handwriting recognition involves reading and interpreting handwritten input from sources such as manuscripts, paper documents, images etc. Off-line handwriting recognition involves converting text from the given image to sequence of characters of a language which can be used by further text processing applications.

The project is end to end detection and recognition of handwritten document. The project is a combination of line Segmentation module followed by word segmentation and then finally detection of words.

Literature Review

For Line Segmentation

Many well proposed method for line segmentation first try to finds a potential candidate as a starting point of lines that separate upper and lower text region. Horizontal projection profiling is used for this. It is the sum of pixel value in any giving direction.

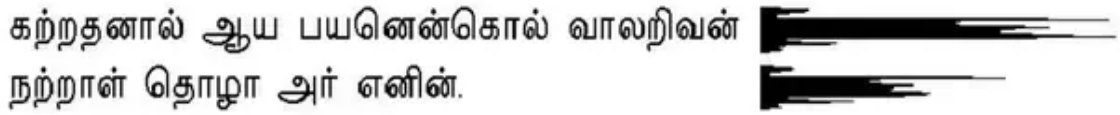


Fig. Horizontal projection profiling

This profiling help in identifying the gap between two lines, thus making them easy to separate.

Method proposed by Bulacu et al. The document is first divided into chunks and then the projection is carried out. The valley obtained in in this way act as a starting point of text line.

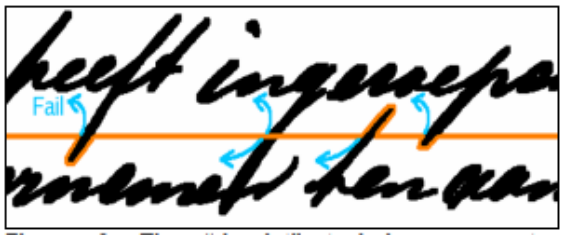


Fig. Water droplet following orange path

Once the starting point is obtained the "droplet" method is used.

Firstly the straight line is created, then the image is turned 90 degree and some artificially drop is allowed to fall through the paper to follow the straight as much as possible. If the boundary (ink) is detected the drop will follow two path (up and down shown as blue arrow in fig). The drop will follow the path that is open. If both the path are closed, means the upper and the lower line are overlapping. In this case the drop is allowed to go over the ink.

Method proposed by Garz et al : In this method, firstly the interest point are detected with the help of Difference of Gaussians. These interest point denotes the significant area in the document. Once this is done, the energy graph is created for the region around interest point and the minimum energy region path leads to line segmentation.

Method proposed by Olarik:

A* path finding: This is used in path finding between two point among grid of points.

The method uses the well-known A* algorithm to segment the document into lines. The A* algorithm is combined with many cost function to determine the optimal line separating the upper and lower text regions. The vivid cost function helped in finding the optimal lines even if the lines are somewhat overlapping.

Steps:

1. Document binarization is done.
2. Once the binarization is done, the horizontal profiling is done and local minima is taken as the starting point of the line. $S = \{0, Y_m\}$ and the end point is taken on the same y level and width of document is taken as x coordinate of end point. $E = \{w, Y_m\}$.
3. Once starting and end point are fixed the A* algorithm is applied. The Ink(text) is considered as block region.

Modification:

1. **When the Overlapping is present**: Since the algorithm consider text region as block region. If upper and lower region are overlapped then the right path could not be found. To eliminate this agent are allowed to move through the text region.

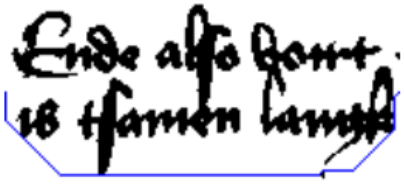


Fig. A* detecting wrong line segment

2. **New Cost Functions**: The modified A* uses two cost function, namely minimal distance cost function and neighboring cost function.

a. **Minimal distance cost function**: Based on this value the cursor moves up or down in the line segmentation. The formula for the same is:

$$D(n) = \frac{C}{1 + min\left(d\left(n_{y_u}\right), d\left(n_{y_d}\right)\right)}$$

Here the C is constant, smaller C value make line more straighter and large value of C allow the cursor to move freely. The d represent distance between agent locations to vertical points. If no block region is found, the max value is used for the same.

Fig. Here the minimum of (ny, y1) and (ny, y2) is chosen for D(n).

b. Neighboring Cost function N(n): The neighboring cost is calculated for all neighbor of current node. Cost 10 is used for vertical and horizontal direction and 14 for diagonal nodes. Since the cursor needs not to go back, only five possible direction are considered.

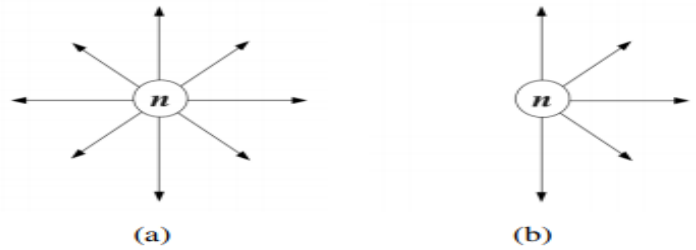


Fig. Neighbor to be considered in **a)** original A* **b)** modified A*

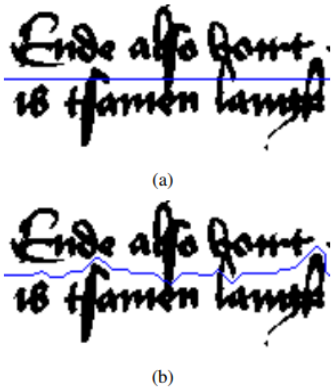


Fig. a) Small value of C **b)** Large value of C

So the new function for the A* is modified as:

F(n) = G'(n) + H(n)

Where G'(n) = D(n) + N(n)

The method produced as accuracy around 97% for Saint Gall dataset.

For Word Segmentation

Scale Space

At a time we want to focus on only one detail of image instead of complete image. This is where scale space comes into picture. Here we take image keep blurring it at certain scale (scale keeps on increasing linearly). Convolution of Gaussian with image is referred as "Blurring". Gaussian gives us Linear Scale space. There is a particular operator for Gaussian blurring.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

1) I is an input image.
2) G is Gaussian blurring operator.
3) L is the blurred image obtained.
4) x, y is co-ordinate locations
5) σ is "scale", the scale with which blurring is applied.

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

Blob Analysis

After Line segmentation we have to examine each word present in the line, word can be formed of distict characters or connected characters. When we merge these all it gives us a meaningful word. This may be achieved by using blob like structure in image.

If we want to find images with particular objects. We first need to find different objects present in the image. For this we have to separate the objects from image and then need to find the objects which we are looking for. The former is called as BLOB EXTRACTION and the later is called as BLOB CLASSIFICATION. "BLOB" stands for Binary large object because only large objects are useful to us as small objects are usually noise.

Laplasian of Gaussian is used frequently for producing blobs in image. But here we will be using a differential expression different from LOG . Using second order differential equation for 2 different orientations.

Spatial extent of word is defined by :-

1) Each character defines the height of word(y-axis).
2) No of characters in word defines length of word(x-axis).

As x-axis is usually larger than y-axis in word instead of going for same scale in both direction use different scale in both directions. Which makes the gaussian operator as

$$G(x, y; \sigma_x, \sigma_y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-(\frac{x^2}{2\sigma_x^2} + \frac{y^2}{2\sigma_y^2})}$$

Scale in x-direction is taken larger compared to scale in y direction corresponding to the word.

$$\eta = \frac{\sigma_x}{\sigma_y}$$

For the above gaussian the second order differential equation is defined as

$$L(x, y; \sigma_x, \sigma_y) = G_{xx}(x, y; \sigma_x, \sigma_y) + G_{yy}(x, y; \sigma_x, \sigma_y)$$

Scale space image is produced by convolving the input image with the gaussian differetial equation defined above

$$I(x, y; \sigma_x, \sigma_y) = L(x, y; \sigma_x, \sigma_y) \star f(x, y)$$

After applying this gaussian filter recurssively at certain scales we will get character blobs, word blobs, line blobs. Fig (a) is line image, after blurring it in small scale it will give us character blobs in Fig (b), after blurring the image with a bit larger scale it gives us image in Fig(b) which is giving word blobs. This gives us that at certain scales we will get the image with eord blobs, after extracting these blobs we can obtain words from line.

(a) A line image

(b) Blob image at scale $\sigma_y = 1, \sigma_x = 2$

(c) Blob image at scale $\sigma_y = 2, \sigma_x = 4$

(d) Blob image at scale $\sigma_y = 4, \sigma_x = 16$

(e) Blob image at scale $\sigma_y = 6, \sigma_x = 36$

Choice of Scale:

Maximum response in the scale is obatained when we use the scale proportional to the object's dimension. One of the advantages in document image is it does not need much variation in scale to obtain images with character, word blobs. There exist a scale at which blobs is formed in distict words we have to get this scale so that word can be obtained from line.

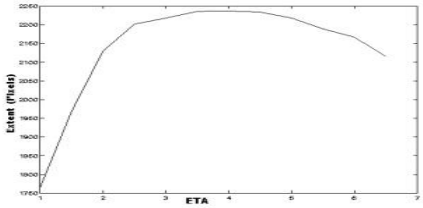
Blobs consists of Spatial Extent. Algorithm requires scaling factor along y-axis and multiplication factor for blob extraction. After analysis it was found that simple sacle selection is based on that maximum of spatial extent of the blobs. That is maximum of spatial extent among blobs present gives us scale. Sum of extent of a line is given by A as shown below.

$$A = \sum_{i=1}^{n} \zeta_i$$

- **Selecting η:**

  After analysis of several images average aspect ratio was found out to lie between 3-5. After keeping keeping scale along the y-axis constant got the maximum spatial extent by taking multiplication factor between 3-5. In the figure shown below maximum is obtained at 3.8. Hence from the observation if aspect ratio is around 3-5 we can choose multiplication factor between 3-5.

(a) A line image



(b) Plot of extent vs η, $\sigma_y = 2$ for above image, a maximum is obtained at $\eta = 3.8$

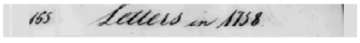
- **Selecting scale along y-axis:**

  After several analysis it is found out that scale along y-axis is dependent on height of characters.

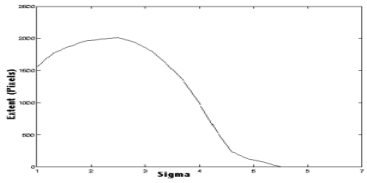
$$\sigma_y = k \times \text{Line height}$$

Where $0 < k < 1$. After experimenting different scales to find the maximum. For the case below consider value of k=0.1 and varying scale along the y-axis with change of 0.3 which worked well for many images considered.

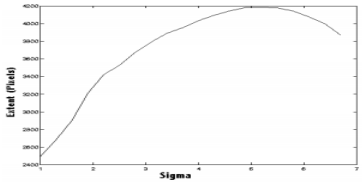
(a) A sample line image with smaller height



(b) A sample line image with larger height



(c) Plot of extent vs σ_y,Maximum is obtained at $\sigma_y = 2.5$



(d) Plot of extent vs σ_y, Maximum is obtained at $\sigma_y = 6.0$

End to End Handwritten Recognition using MDLSTM:

In this method feature extraction take place after scanning input image in different direction. After feature extraction the characters are recognized by passing feature map to decoder.

The advantage of this model is that it is first end to end model for handwritten paragraph recognition as segmentation and classification is taking place in the same model itself. It do not require line segmentation and word segmentation.

The drawback of this model is it is detecting character using only visual classification of characters instead of using language model (which considers context of word to detect each character).

Handwritten Text Recognition using Deep Learning

Pre-processing:

Before feeding the input images in the model we do some data augmentation and pre-processing to make the training data more robust.

1) Padding image
2) Rotating image
3) Zero Centering data

Padding: Architecture here assumed the word images of same size (same width and height). Because each word is of different size and when fed into CNN weights in the first layers is based on input data, which will result in inconsistent weights. Instead of cropping the image or resizing it, will do padding to make it of specific size so that no data will be loss.
Rotating image: Image is rotated right by small angle to make the training data more robust.
Zero Centering data: We center the image by subtracting the image pixel with mean of all the pixel values.

Methods:

Word Level Classification Model:

First method used VGG-19 as architecture, but it was too time consuming due to large amount of data available in word vocabulary, to reduce the training time, training was done with data of 50 random words which appears for at least 20 times in dataset. Still the training was time consuming.

In second method RESNET-18 was used which gave almost same accuracy as that of VGG but was much faster compared to VGG-19, and the residual network of RESNET-18. Since the number of epochs was low with many different hyper-parameters, using only optimal number of hyper-parameters with more number of epoch which made accuracy better. After using RESNET-34 accuracy increased further.

But to obtain better accuracy instead of mapping the features to large amount of word dataset the word are segmented into characters and its features are mapped to character which is a comparatively much smaller set.

Character Level Classification:

Instead of using word level classifier, used character level classifier. The model uses CTC RNN based model. CTC is method which produces probability distribution for output sequences from the given un segmented input. CTC along with RNN found out to be better approach as compared to traditional approaches. It can be seen from the table below that mapping features to character list instead of word dictionary gives better accuracy.

Architecture	Training Accuracy	Validation Accuracy	Test Accuracy
VGG-19	28%	22%	20%
RESNET-18	31%	23%	22%
RESNET-34	35%	27%	25%
Char-Level	38%	33%	31%

Drawback:

As model was not trained on large no of dataset it gave imperfect segmentation. Model also finds difficulty in segmenting cursive characters because of breakdown in boundaries. Also because of different styles of handwriting of different sizes makes it difficult to recognize characters/words.

Limitation of existing Models

- In End to End Handwritten Recognition using MDLSTM model, it is detecting character using only visual classification of characters instead of using language model (which considers context of word to detect each character).
- Handwritten Text Recognition using Deep Learning is for the detection of Words only i.e. in one go only a word can be detected. Thus it is not an end to end model. The input document need to be segmented into line and then words before this architecture can be used.
- The model uses the greedy search for recognition of word i.e. Best Path encoding. This leads to somewhat wrong detection of words. Instead of this the better way is to use dictionary to output the word (from dictionary) closest to nearest recognized word.

Dataset Used

We have used IAM dataset for the training the testing purpose. The database contains handwritten forms, lines, sentences and words. The writing is from different writers (500+). 1500+ forms, around 115000+ words along with their label. The words are segmented and verified manually corresponding to ground truth values. We have used only words to train our model.

Proposed Method

Model architecture

End to end detection and recognition of handwritten document consists of following steps:

Preprocessing Step

1. Binarization
2. Line Segmentation
3. Word Segmentation
4. Input Resizing and gray scale conversion

Recognition

1. Feature Extraction through CNN.
2. Feature Mapping through RNN.
3. Decoding text using CTC.

Preprocessing

Binarization

The process of separating image foreground from page background during document image analysis is called Image Binarization. It removes stains or some faded ink marks from the background which helps in not only analyzing document but improves document readability as well.

For the Binarization purpose we have used Sauvola's adaptive document image binarization.

Line Segmentation

Given a handwritten document, to detect the words present in the handwritten document from the model we have created, the first step is to identify the region of words and before the words can be recognized, firstly line need to be determined of which the particular word is part of. This process of segmenting the document into various line is referred as Line segmentation.

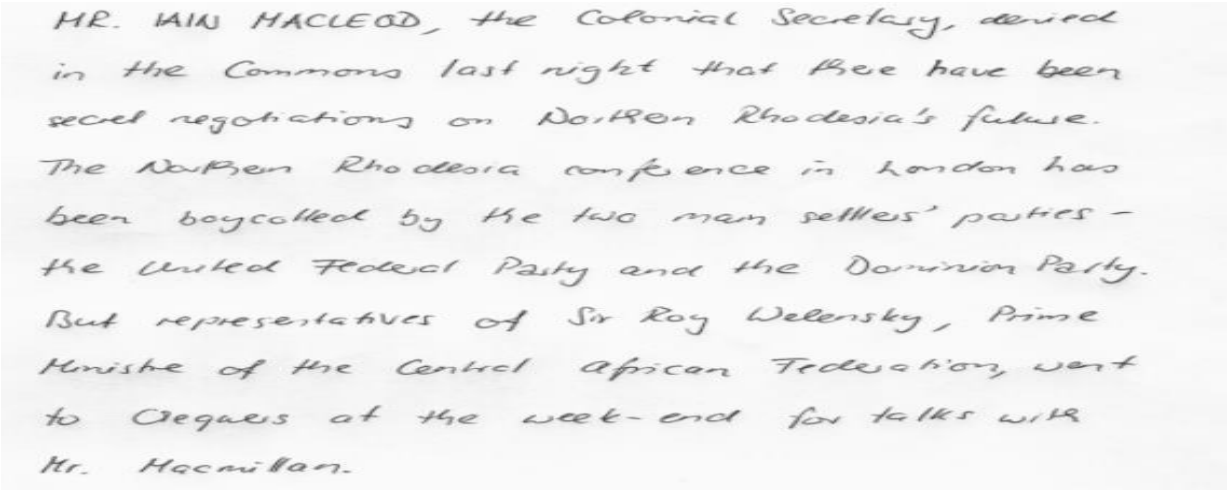


Fig. Input Document

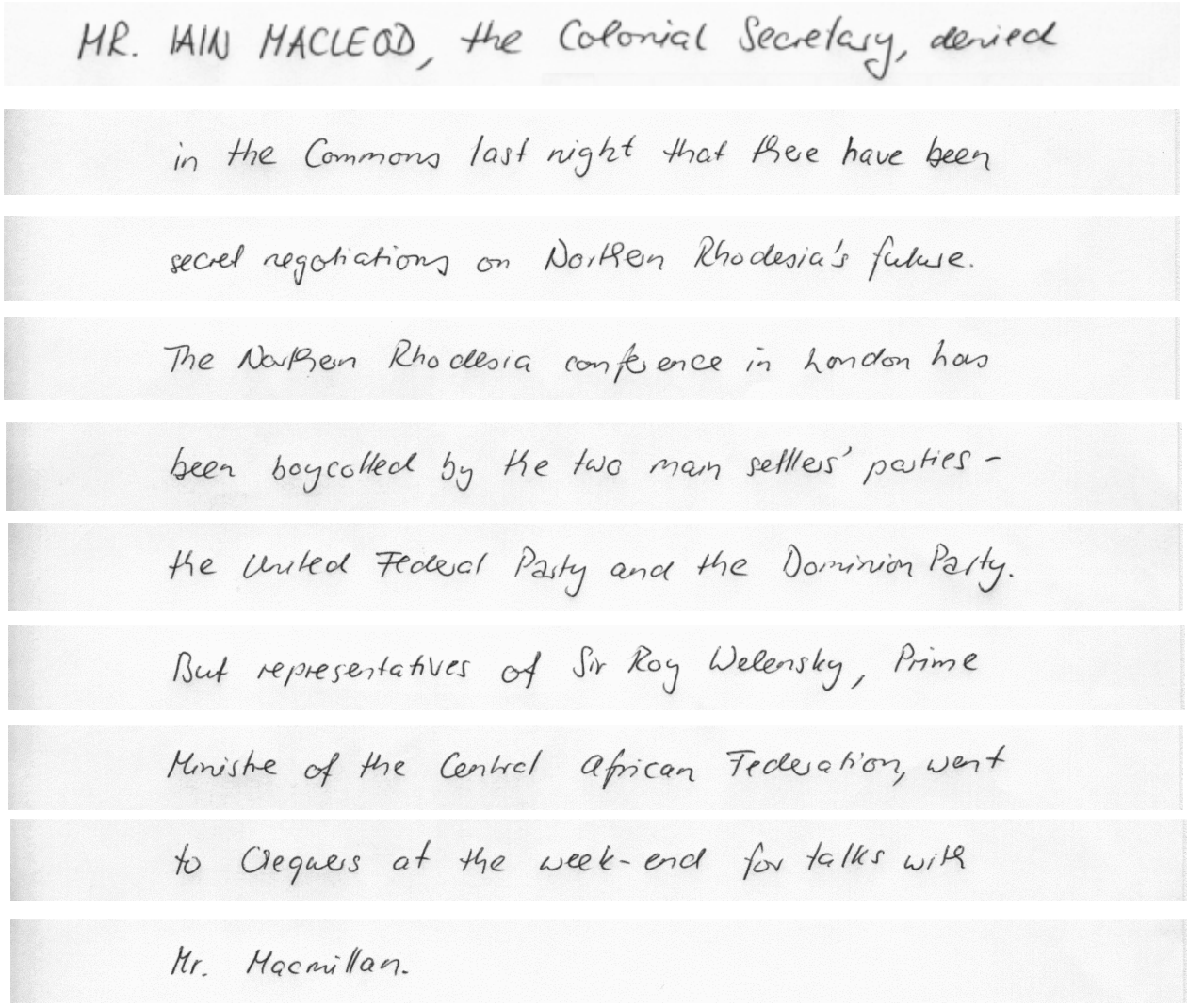


Fig. Output of Line Segmentation

Used Method

Input: A Handwritten document. (We have taken forms and documents from IAM dataset)
Output: Document with separating lines and each line is stored cropped and stored separately. So that this can be fed to Word Segmentation directly. As shown in above Figure.

For this we have used the A* path finding Algorithm explained above.

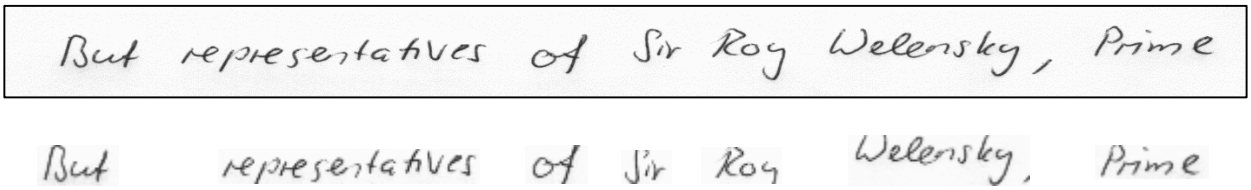
Word Segmentation

After doing line segmentation on the handwritten document, we need to find word boundaries in each line, so that we can split the lines into meaningful words. Word segmentation is a process of finding meaningful words from a sentence or document.

Doing word segmentation on handwritten text document is complex because:-

1) Each character is written in distinct way.
2) There is not uniform spacing in handwritten text.
3) Scale problem (characters are of different size).

Input and Output



For this we have used method based on scale space technique described above.

Input Resizing and Gray Scale conversion

The image obtained after word Segmentation may not necessarily be in desired size. That is why the image is first resized to either width of 128 or a height of 32 and then padding is done. After this Image is normalized for gray scale value.

Recognition

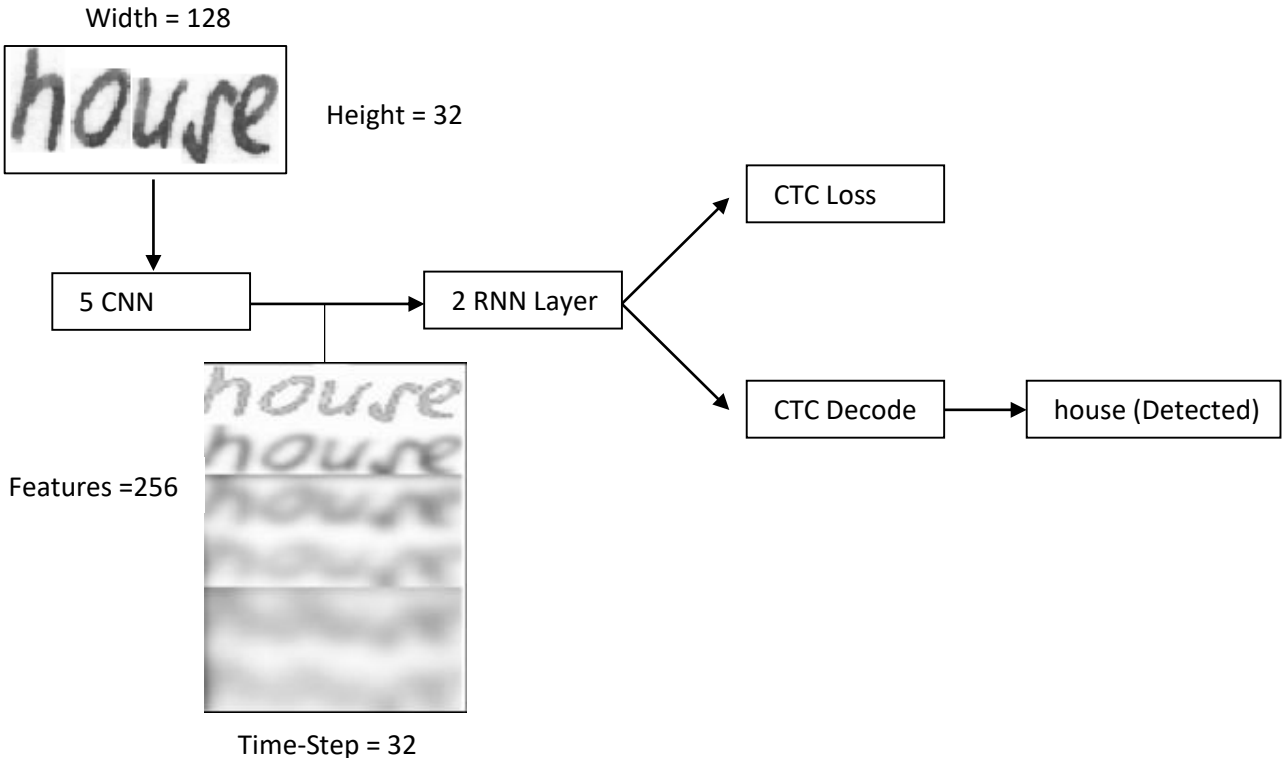


Fig. A High level Architecture of Word

Feature Extraction through CNN

1. After preprocessing the input image is 128 X 32.
2. It is fed into 5 layer of CNN which gives the feature map of size 32 x 256.

Three operation are performed on each layer:

i. Convolution
ii. Relu is applied
iii. Max pooling.

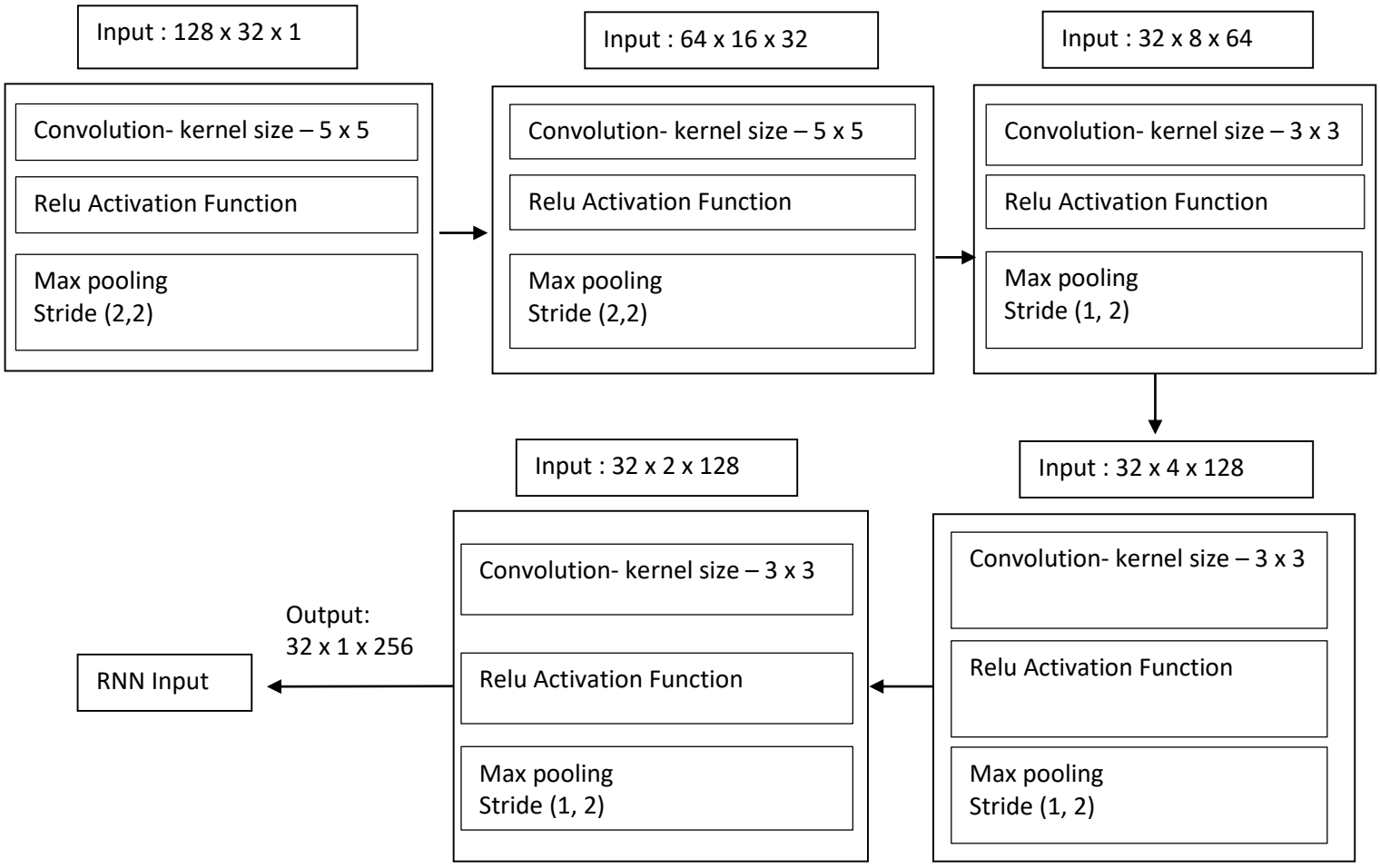


Fig. 5 Layer Architecture of Convolution Neural Network Architecture

Feature Mapping through RNN

The feature map obtained from CNN is of size 32 x 256. RNN propagates useful information(character-scores) through this sequence and map it to a matrix of size 32 x 80. We have used bidirectional RNN using LSTM cell because information can be propagated through longer distance. The IAM dataset consists of 80 different characters that is why each time step (32 in no.) is mapped to 80 entries.

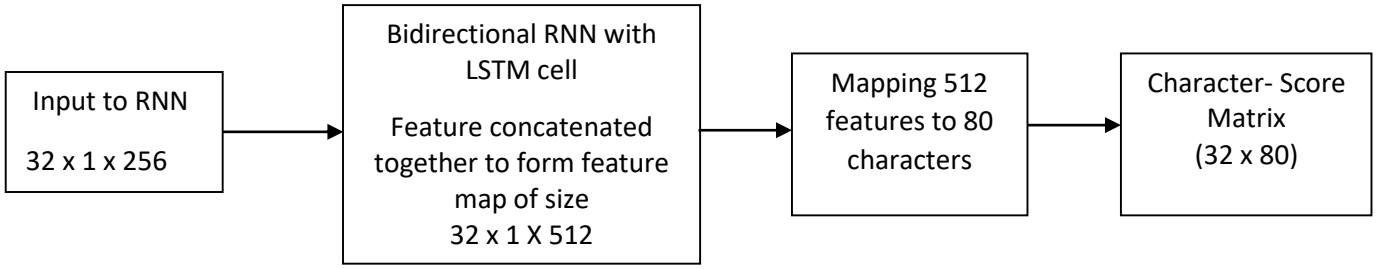


Fig. Flow through RNN layers

Decoding text using CTC

The output of RNN is character score matrix. This is fed to CTC along with the ground truth value of text. There are two main operation of CTC.

1. Train – Loss value calculation to train Neural Network by trying all possible alignment of character sequence.
2. Infer – Decode RNN output to get the final text using Best Path Decoding or Word beam Search.

Best Path Decoding: It uses the output generated by the Neural Network

1. Take the most likely character at every time step.
2. Undoes the encoding by removing duplicate and blank space.

Beam Search: It explores a graph(remaining output) by expanding only the most promising node. Thus at each step most likely output are explored. Beam search allow arbitrary character to get detected.

Token Passing: It actually uses the dictionary and searches for the most probable sequence of words from dictionary. Main disadvantage is it cannot handle arbitrary sequences like number, punctuators and words are predicted from dictionary only. This avoid spelling mistakes.

Word Beam Search: It uses the advantage of Token passing and Beam search. Whenever the special character are detected, beam search is used otherwise Token passing technique is used. This help in removing the disadvantages of Token passing and Beam search.

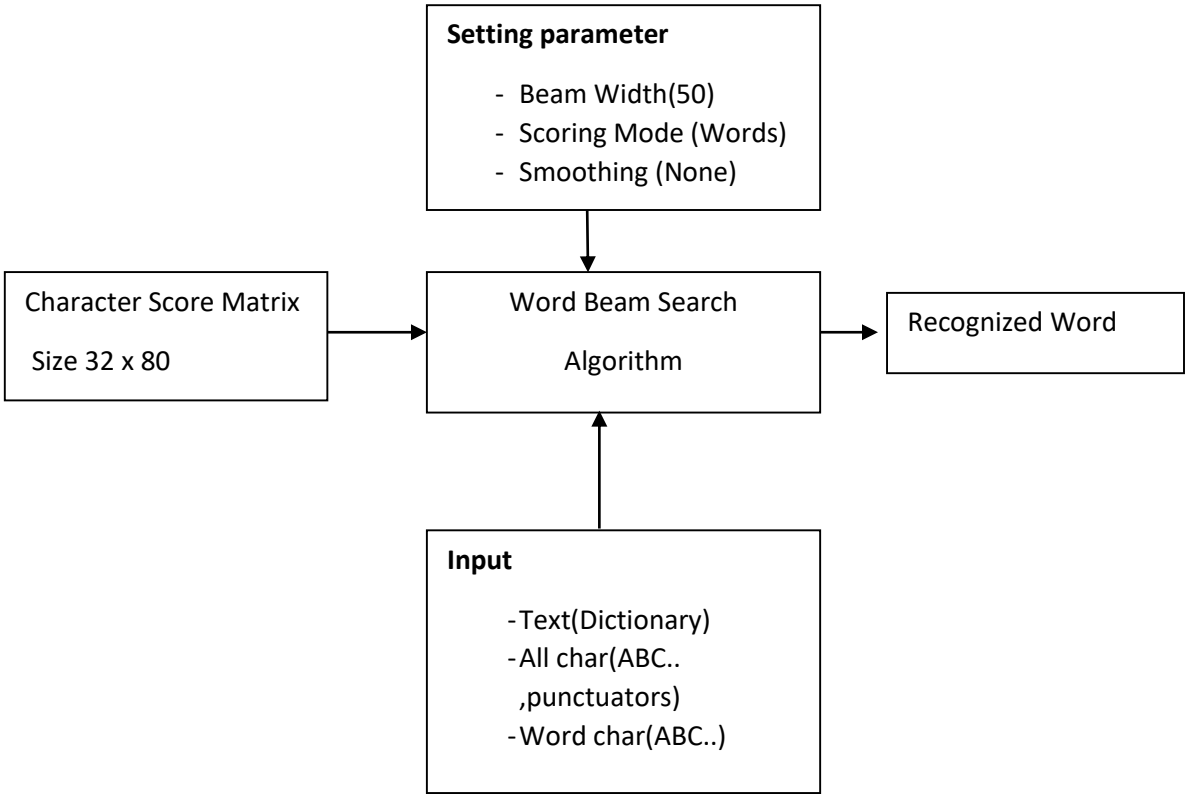


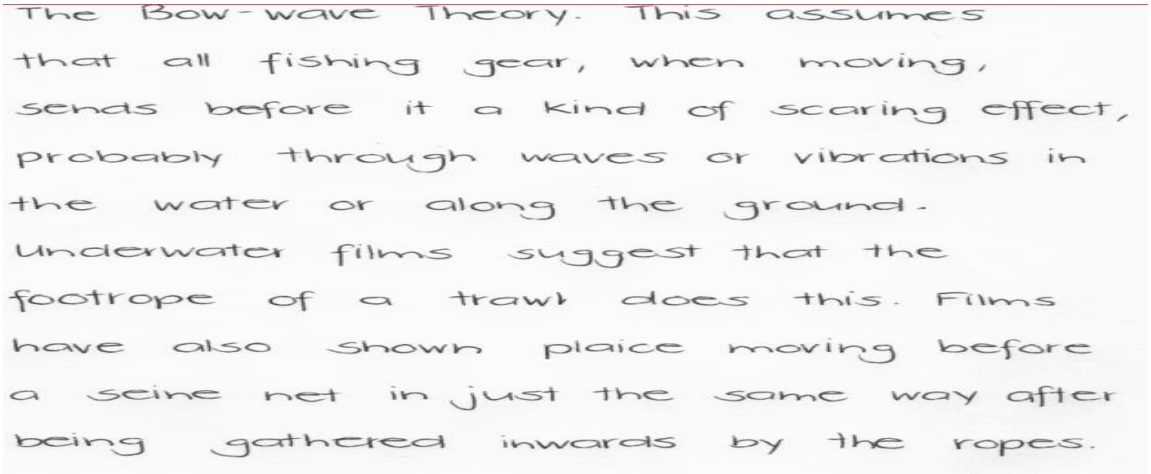
Fig. Block Diagram showing working of Word Beam Search

Results:

For input form image from IAM dataset following results are obtained as output from our recognition system:

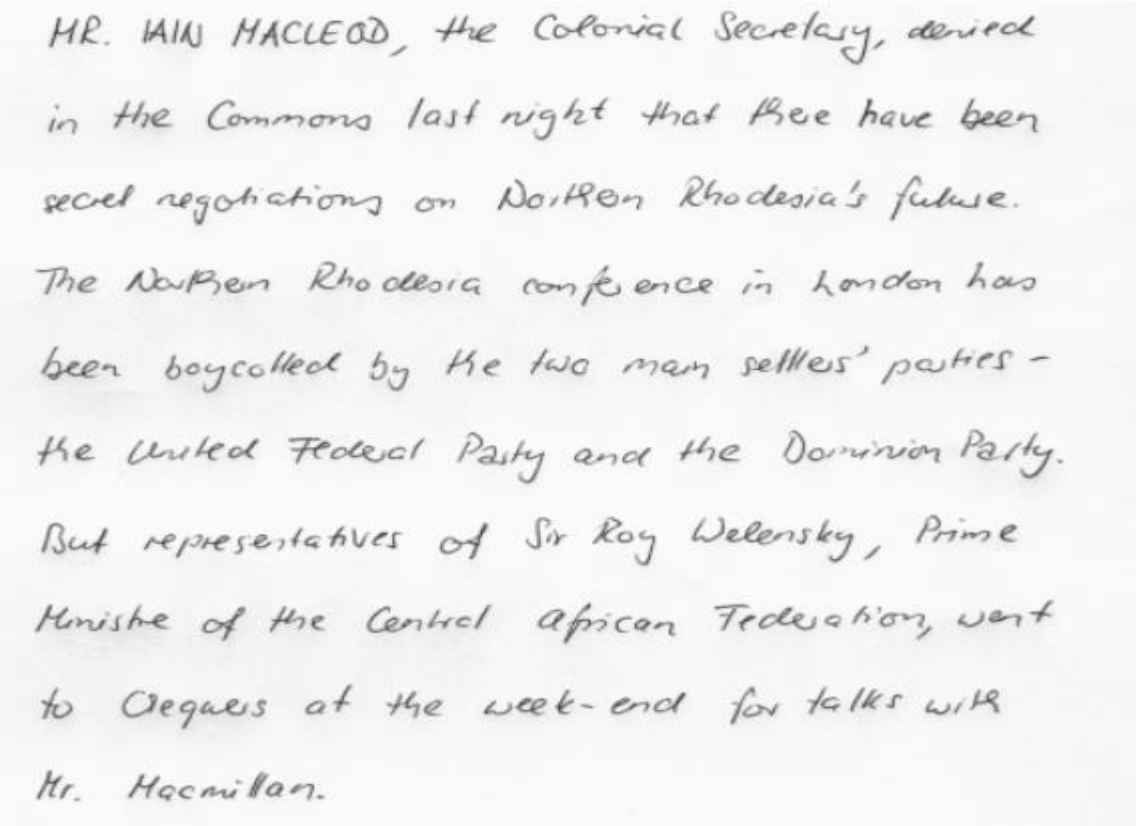
The document is first segmented in lines followed by word segmentation. The result obtained by (Number of word detected correctly) using various decoder are stated below:

Image1:



Line Number	Number of words in line	Best Path	Beam Search	WordBeam Search
1	6	5	6	6
2	6	6	6	6
3	8	5	6	6
4	6	5	5	6
5	5	5	5	6
6	7	6	6	7
7	6	6	6	6
8	9	7	7	8
Total number of words	53	45	47	51
Accuracy		**84.9%**	**88.67%**	**96.22%**

Image2:



Line Number	Number of words in line	Best Path	Beam Search	Word Beam Search
1	9	4	4	4
2	6	3	3	5
3	7	1	1	4
4	8	6	6	8
5	8	2	2	4
6	7	6	6	6
7	7	3	3	4
8	9	7	7	9
Total number of words	61	32	32	44
Accuracy		52.4%	52.4%	72.13%

Since our implementation is segmenting form into words and we use ground truth value of word instead of form, the accuracy is calculated manually for different forms from IAM Dataset.

Conclusion

In this report we have proposed a method to convert the handwritten document into digital text. Firstly the document is segmented into lines, for this we have used A* path finding algorithm with modified cost function. Secondly the lines are segmented into word by analyzing extent of blob using scale space. The segmented words are fed to CNN followed by RNN and CTC to recognize the words. Further we have used decoding algorithm word beam search that uses language model and dictionary to improve accuracy of the recognized word and also it allow non-arbitrary character like numbers or punctuator marks. Also we have analyzed various architectures, preprocessing steps to check for accuracy.