Neural Network based Cursive Handwriting Recognition

Neeta Nain, Ankit Agarwal, Anshul Tak, Gaurav Jain

***Abstract*—In this paper, we have proposed a novel approach for handwriting recognition system involving segmentation for preprocessing steps and using diagonal based feature extraction technique with neutral network for character recognition. Input is paragraphs of running text, which is preprocessed to segment it into normalized individual words. Further, a diagonal based feature extraction technique is used for extracting the features of handwritten alphabets. A neural network is trained onto the dataset containing 55 samples for each of the 26 alphabets for recognition. A new implicit approach for character recognition is implemented in this paper which segments a character, into parts dynamically for character recognition from the text, which improves the accuracy significantly. A feed forward artificial neural network is being used for character classification, which also helps in deciding the threshold value for the character separation from the running text word. The proposed recognition system performs excellently for separate character written documents with 100% accuracy. It also performs competitively yielding an accuracy of 75% for readable non-cursive handwriting and 60% for cursive handwriting.**

***Index Terms*—Handwriting character recognition, character segmentation, feature extraction, preprocessing, feed forward neural network.**

I. Introduction

Handwriting recognition has been one of the most fascinating and challenging research areas in field of image processing and pattern recognition in the recent years. Offline recognition is performed on a scanned image of handwriting and thus contains no temporal data. In general, handwriting recognition is classified into two types as off-line and on-line handwriting recognition methods. In the off-line recognition, the writing is usually captured optically by a scanner and the full text is available as an image therefore it contains no temporal data. Some existing techniques are fusion based segmentation method [2]. In this approach, over segmentation of words from text based on pixel density between upper and lower base line with multiple expert base validation for character recognition and classification has been developed. Slant and skew errors are neglected in this approach. In K-nearest neighbor [4] technique, feature extraction depends on Euclidean distance between testing point and reference point. Which is used to calculate KNN neighbor. This method could classify images containing single characters. In N-gram [1], training is done on text corpus and character can be recognized belonging to this corpus only. We propose a system which could extract characters from running text. This has been attained by addressing all the individual steps of character recognition like: binarization, line segmentation, skew correction, slant correction, baseline positioning, to individual character recognition.

The authors are with Malaviya National Institute of Technology Jaipur, JLN Marg Jaipur- 302017, Rajasthan, India
(email: neetanain@yahoo.com)

In the on-line system the two dimensional coordinates of successive points are represented as a function of time and the order of strokes made by the writer are also available. Some previous known algorithms are: Elastic structure matching [3] approach which recognize online handwriting, it works best on digits only and manual training is required on characters. The normalized and rectification method [5], is used to obtain the best matched of the character but this can not separate similar looking digits.

Section II explains (a) preprocessing steps: line segmentation, skew and slant correction, word segmentation etc in subsection II-A. It segments scanned text image into individual words; (b) subsection II-B explains the proposed technique for feature extraction, classification, and recognition using Artificial Neural Network (ANN). Section III does test and result analysis. It describes the various test cases and their accuracies, finally Section IV summarizes the findings.

II. IMPLEMENTATION PROCESS

For handwriting recognition the input to our system is a scanned image containing handwritten text paragraph. The image should be in prescribed format such as jpeg, png etc. An input sample image is shown in Figure 1

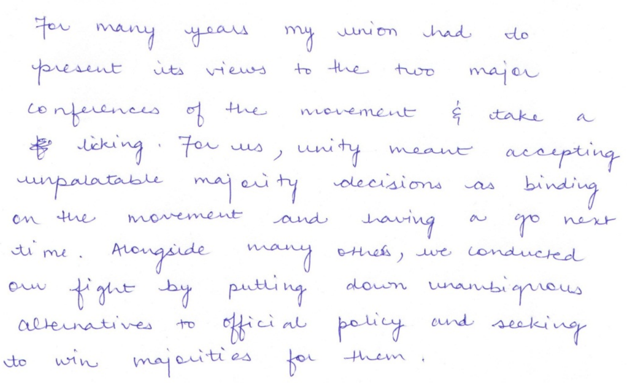

For many years my union had to
present its views to the two major
conferences of the movement & take a
~~[illegible]~~ licking. For us, unity meant accepting
unpalatable majority decisions as binding
on the movement and having a go next
time. Alongside many others, we conducted
our fight by putting down unambiguous
alternatives to official policy and seeking
to win majorities for them.

Fig. 1: Example input scanned text image.

A. **Preprocessing**

Preprocessing is the major step in handwriting recognition system. It employs several step, that is, line segmentation, skew correction, baseline correction etc. It takes input as a raw running text image and gives output as segmented words.

The foundation of our preprocessing steps is based on highly cited paper [8]. Figure 2 shows the schematic diagram of the preprocessing steps.

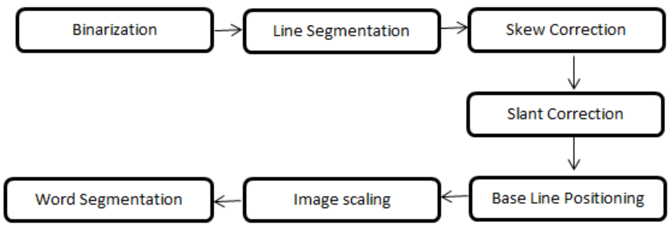


Fig. 2: The schematic diagram of preprocessing.

1) **Image Binarization:** Binarization is done by converting an image from its input color map i.e. RGB to gray scale and then converting it into black and white using a threshold calculated by Equation 1. We exhaustively search for threshold (t) that minimizes the intra-class variance (σ_i^2) defined as awaited sum of variance of two classes

$$\sigma_\omega^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t) \qquad (1)$$

where weights ω_i are the probabilities of the two classes separated by threshold t and σ_i^2 is variance of these classes. Minimizing the intra-class variance is same as maximizing inter-class variance as

$$\sigma_b^2(t) = \sigma^2 - \sigma_\omega^2(t) = \omega_1(t)\omega_2(t)[\mu_1(t) - \mu_2(t)]^2$$

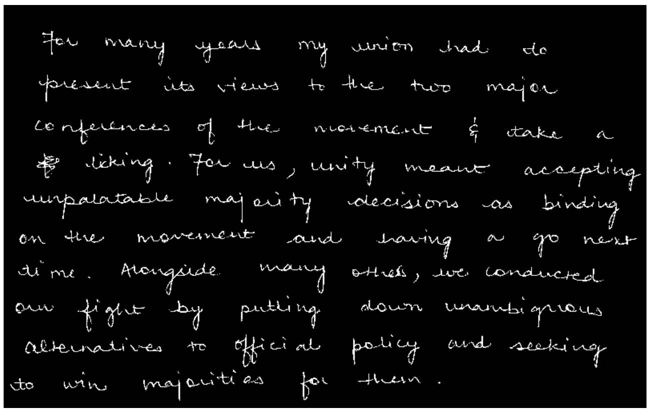
which is expressed in terms of class probabilities ω_i and class mean μ_i. The class probability $\omega_1(t)$ is computed from the histogram as t

$$\omega_1(t) = \Sigma_0^t p(i)$$

while the class mean $\mu_1(t)$ is

$$\mu_1(t) = \Sigma_0^t p(i)x(i)$$

where $x(i)$ is the value at the center of i^{th} histogram bin. Similarly, $\omega_2(t)$ and μ_t can be computed on the right-hand side of the histogram for bins greater than t.



For many years my union had to present its views to the two major conferences of the movement & take a liking. For us, unity meant accepting unpalatable majority decisions as binding on the movement and having a go next time. Alongside many others, we conducted our fight by putting down unambiguous alternatives to official policy and seeking to win majorities for them.

Fig. 3: Result of binarization step on Figure 1.

2) **Line Segmentation:** In this step, lines are segmented from paragraph [9] assuming that lines are relatively horizontal. Line segmentation involves the following steps

1) A horizontal histogram is made for the binarized image as shown in Figure 4.

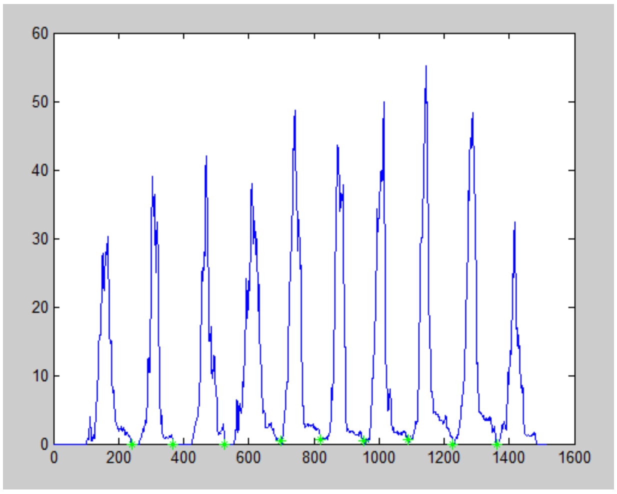


Fig. 4: Histogram and its minima.

2) Proper minimas and maximas of histogram are detected with median filter as a smoothing function for the given histogram as shown in green color in Figure 4. Here $1D$ median filter is used for smoothing. The median filter is an effective method that can suppress isolated noise without blurring sharp edges. Specifically, the median filter replaces a pixel by the median of all pixels in the neighborhood as explained in Equation 2.

$$y[m,n] = \{median(x[i,j],(i,j) \in w)\} \qquad (2)$$

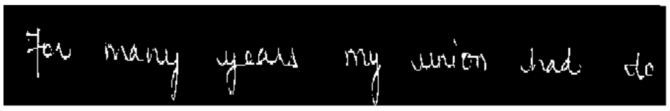
where w represents a neighborhood centered around location (m,n) in the image.

3) Cuts are made at significant minima. A minima of 0 value signifies there exist no intersecting ascender and descenders at the cut $Cut[x_i] = \{minima(vhist(i,j))\}(i,j) \in U$.

4) IF there exist an intersection at minima lines on either side of the cut, it is further analyzed for connected component. Further, centroid(C) of the connected component(CC) is calculated to classify it is ascender or descender. Three cases for CC are identified

$$C \in \begin{cases} Cut[i-1] + D \leq CC \leq Cut[i] - D & \in \text{current line} \\ < Cut[i-1] + D & \in \text{previous line} \\ \text{else} & \in \text{next line} \end{cases}$$

where threshold (D) is $\frac{1}{8}$ part of the difference of current minima and previous minima.



For many years my union had to

Fig. 5: First segmented line (after line segmentation process) of Figure 1.

Figure 5, shows first segmented line using the above steps.

3) **Skew correction:** Majority of people fails to write straight lines in their conventional style leading to a skew error in line. This is resolved by skew correction as: Compute the count of foreground pixels in a vector Q using Equation 3.

$$Q = \{q_i = (x_i, y_i) | \text{Lowest black pixel} \in x_i \text{ column}\} \qquad (3)$$

Q_i represents a vector of bottom most pixels of each line for the i^{th} column.
The skew is computed by fitting a straight line $(L) = y - mx - c$ from the vector Q using least square sense. The slope (m), of the line (L) gives the skew angle θ as: $\tan^{-1}(m)$. To normalize the line segment, for skew error, rotate it by -θ. A line with skew error, and its normalization is shown in Figures 6 and 7, respectively.

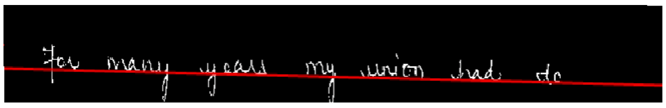


Fig. 6: Line segment before skew correction.

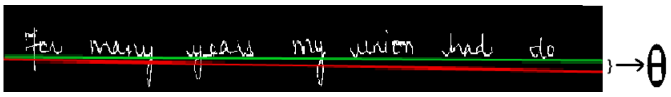


Fig. 7: Line after skew error is removed by a rotation of angle θ.

4) **Slant correction:** Slant correction [6] is used to normalize the text writing (when the writers text is at an angle with the normal to the baseline), making the text upright. This is a two step process

1) Rotate the text line from $-45°$to $45°$(italic slant error range), and calculate vertical histogram for each angle of rotation.
2) Compute Wigner-Ville [13] distribution of each histogram and the angle with largest distribution intensity, is taken as our slant angle α.

Wigner-Ville distribution [13] is related to time frequency representation of non-stationary signals. In our case, it is affine transformation of an image on intervals of 5^o. For a signal, $s(t)$ with analytic associate $x(t)$, the Wigner-Ville distribution $W_x(t,\omega)$ is defined as

$$W_x(t,\omega) = \int_{-\infty}^{\infty} x(t+\tau/2)x^*(t-\tau/2)e^{-i\omega\tau}d\tau$$

where, the analytic associate $x(t)$ of a signal $s(t)$ is defined as $x(t) = s(t) + \iota H[s(t)]$, where $H[s(t)]$ is the Hilbert transform [13] of the signal $s(t)$. Significance of peaks and troughs are deduced and compared by Wigner-Ville distribution. Figure 8 shows slant correction by angle

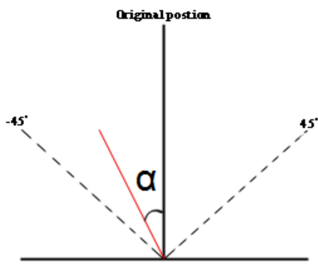


Fig. 8: Red line shows the position where Wigner-Ville distribution intensity is largest.

α. To reduce our computational work and time we have generated transformation for every 5^o angle (step size). The loss of accuracy is not perceptibly significant but it improves the efficiency 5 times.

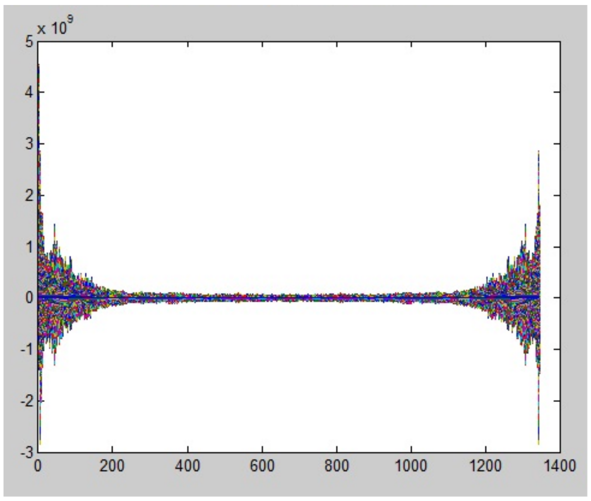


Fig. 9: Wigner ville distribution intensity curve at any random angle.

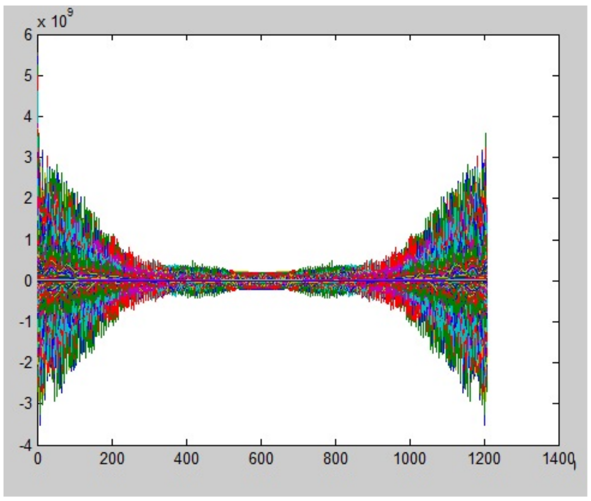


Fig. 10: Wigner ville distribution intensity curve for slant angle α.

5) **Baseline Positioning:** Each line segment is further partitioned into four baselines: Lower Bound (LB), Upper Bound(UB), Lower Baseline(LBL), and Upper Baseline(UBL), defined below. Lower and upper bounds are simple positions where the text starts and ends.

These points can be identified by analyzing each row for first black pixel from top to bottom and from bottom to up. Lower baseline is same as the one located in the skew correction step after rotating by skew angle(θ).

The best position for UBL is typically along the steepest part of the most significant rise in the projection histogram (H_{hist}). To calculate this point, a smoothed version of (H_{hist}) of the segmented line image is analyzed as

1) The derivative is computed and the point with the maximum derivative is chosen: $max_x = \frac{dy}{dx}(H_{hist})$
2) From this point, max_x (maximum slope), the algorithm iteratively expands above and below until the slope falls below a given threshold $(slope_{min})$.

3) UB is the first row which contain first foreground pixel.
4) similarly, LB is the last row which contains foreground pixel.
5) $UBL[i,j]$ points are calculated as:

$$UBL[i,j] = \frac{(lo + hi)}{2}$$

where, $\frac{dy}{dx}H_{hist} > slope_{min} \forall i \in [lo, hi]$.
6) Now UBL can be expressed as $L_{UBL} = y - mx - c$ (in slope intercept form).
7) LBL has already been computed in subsection skew-detection II-A3.

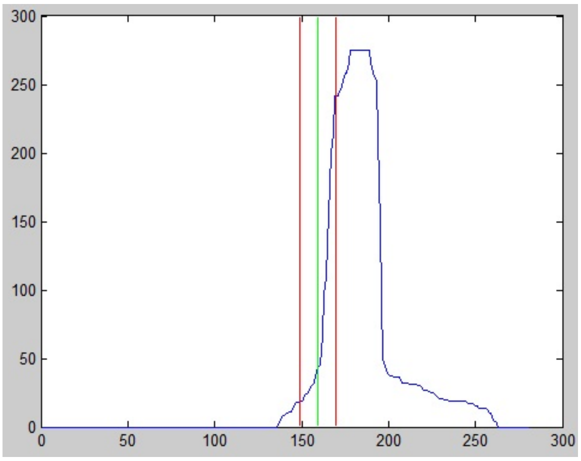


Fig. 11: Red lines represent upper and lower extreme beyond which gradient is less than $slope_{min}$, and green line represents UBL.

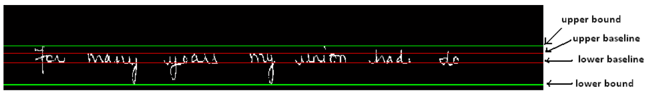


Fig. 12: Baseline positioning.

Figures 11 depicts computation of UBL, and 12 shows all the bounds for baseline positioning.

6) Word segmentation: Word segmentation process is not that easy as it looks because the gaps between the words are never predictive. Sometimes gaps between characters are larger than gaps between words. So to segment [12] word a robust method should be incorporated. The steps for this are

$$V_{hist} = \Sigma_0^i \Sigma_0^j \frac{I(i,j)}{n} \tag{4}$$

First of all a vertical histogram is computed for a line using Equation 4. Then we locate $minima(V_{hist})$ as a prospective word segmentation point. The segmented line may still hold some inappropriate word segmentation points. To clear these faulty points, k - means clustering is performed on the data with $k = 2$.

$$Start_{pt}(i) = \begin{cases} 1 & (V_{hist}(i) == 0 \&\& V_{hist}(i+1) > 0) \\ 0 & otherwise \end{cases} \tag{5}$$

Equation 5 is used to calculate the start points of all words stored in vector $Start_{pt}$, and Equation 6 is used to calculate the end point vector $Stop_{pt}$ of all corresponding words.

$$Stop_{pt}(i) = \begin{cases} 1 & (V_{hist}(i+1) == 0 \&\& V_{hist}(i) > 0) \\ 0 & otherwise \end{cases} \tag{6}$$

Based on these vectors $\{Start_{pt}, Stop_{pt}\}$, we segment the words as shown in Figure 13.

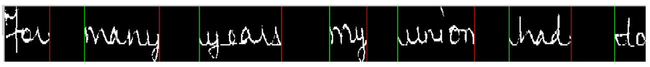


Fig. 13: Word segmentation points.

B. Character Segmentation and Recognition

According to Sayres Paradox [10] a letter cannot be segmented before having been recognized and cannot be recognized before having been segmented. So segmentation, and recognition should work parallel with each other. In the segmentation stage, an image of each word received from previous step is decomposed into sub-images of individual characters.

Characters can be segmented by two approaches

1) **Heuristic based segmentation** [7] - In this approach, stroke width and height estimation are analyzed to segment words into its characters. Then character segmentation region is identified which computes the segmentation boundary between the connected characters.
2) **Neural network approach**- In this approach Artificial Neural Network (ANN) is trained for character recognition. For both training and testing phases, a heuristic feature detection algorithm is used to locate prospective segmentation points in handwritten words.

We have used diagonal based feature extraction technique (explained below), and (ANN) is used for further classification of characters.

1) **Diagonal based Feature Extraction:** In this step, the features of characters that are crucial for classifying them at recognition stage are extracted. We have used diagonal based feature extraction technique [11] in our system. In this algorithm every character image of size 90x60 pixels is divided into 54 equal zones, each of size 10x10 pixels (Figure 14). The features are extracted from each zone pixels by moving along the diagonals of its respective 10x10 pixels. Each zone has 19 diagonal lines and the foreground pixels present along each diagonal line is summed to get a single sub-feature, thus 19 sub-features are obtained from each zone. These 19 sub-feature values are averaged to form a single feature value and placed in the corresponding zone (Figure 14). This procedure is sequentially repeated for the all the zones.

There could be some zones, whose diagonals are empty of foreground pixels. The feature values corresponding to these zones are zero. Finally, 54 features are extracted for each character. In addition, 9 and 6 features are obtained by averaging the values placed in zones row wise and column wise, respectively. As a result, every character is represented by 69 (i.e $54 + 15$) features. The complete step is illustrated in Figure 14.

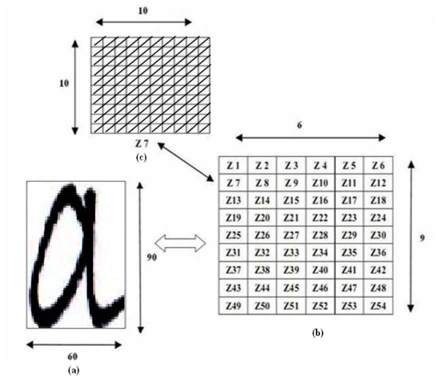


Fig. 14: Procedure for feature extraction from the character.

2) **Training of Neural Network:** We have used two layer feed forward ANN for further classification and recognition of our characters. We have taken 55 sample images for each of the 26 alphabets making it 1430 samples total. Steps taken before inserting data into ANN are as follows

1) Create a window joining leftmost, topmost, rightmost and bottommost pixel of the character in the same order and then this window is cropped from the input image sample.
2) Resize the cropped window to 90x60.
3) Compute the skeleton of the resized image.

Figure 15 illustrated the above steps.

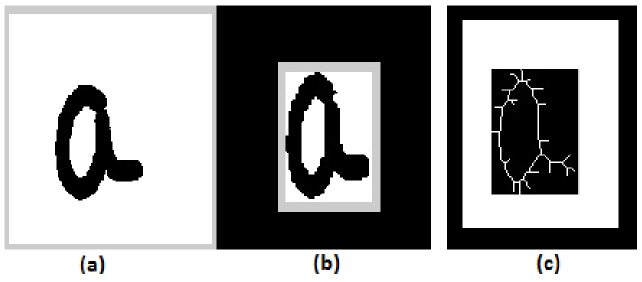


Fig. 15: (a)Input image, (b) cropped $\longrightarrow$ resized image, and (c) Skeleton of the resized image.

Now the above mentioned feature extraction technique is implemented on the skeletonized image which gives 69 feature points for each image accordingly. This procedure is repeated for 26x55 samples and then stored in a feature vector of size 69x1430 which we have taken as input to our ANN as shown in Figure 16. Our the target output for

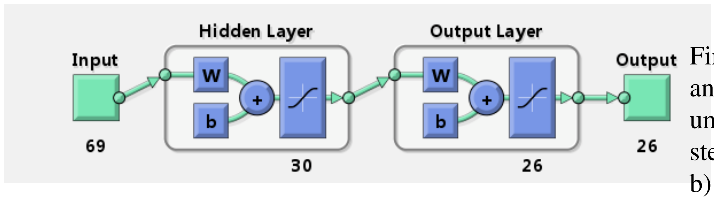


Fig. 16: Artificial Neural Network (ANN).

ANN is 26x1430, where each row represents a character. Taking this as an input and target output we have trained feed forward neural network having a single hidden layer with architecture of 69-30-26. Further specifications are mentioned in Table I.

TABLE I: Specification of ANN.

Input nodes	69
Hidden nodes	30
Output nodes	26
Training algorithm	Scaled conjugate gradient back-proportion
Perform function	Mean square error
Training epochs	1000

3) **Proposed Technique for Recognition:** Characters from the segmented words, received after the completion of pre-processing step are recognized in this step. Various techniques like [14], [1], [4] etc., address character segmentation in parts. Also, for cursive handwriting full character segmentation is further more complicated as the characters are mostly closely connected making perfect segmentation point detection very difficult. For these reasons we propose our own character segmentation technique.

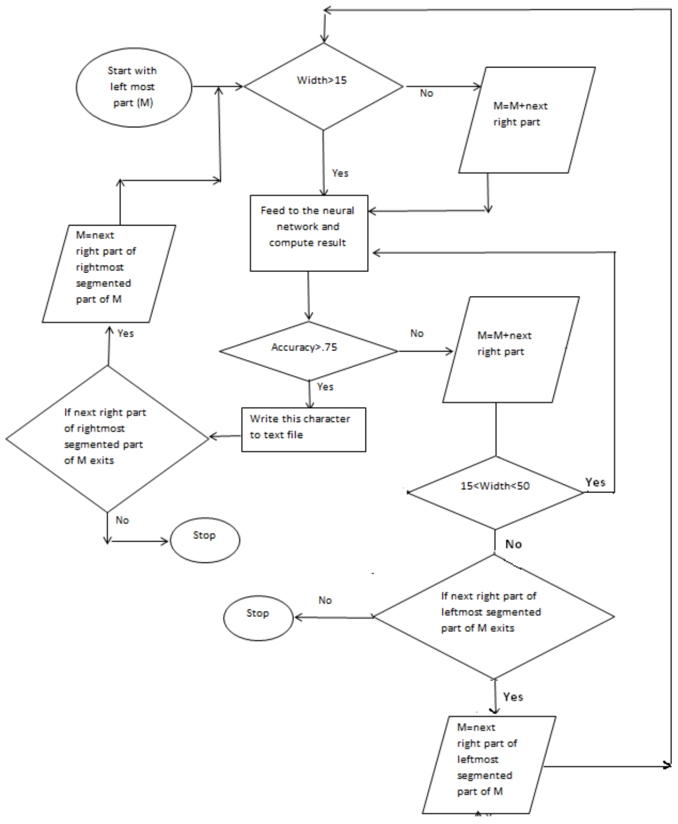

Start with left most part (M)
Width>15
No
M=M+next right part
Yes
Feed to the neural network and compute result
M=next right part of rightmost segmented part of M
Accuracy>.75
No
M=M+next right part
Yes
Yes
Write this character to text file
If next right part of rightmost segmented part of M exits
No
Stop
15<Width<50
Yes
No
If next right part of leftmost segmented part of M exits
No
Stop
Yes
M=next right part of leftmost segmented part of M

Fig. 17: Flow chart for character recognition.

In our implementation we have taken two assumptions. First, there must exists a minima between two characters and secondly, the character width is between 15 to 50 pixels unrestrained to different writers. For word recognition, the steps are: a) compute vertical histogram for each word; b) compute minimas of the histogram and label them as prospective segmentation points; c) divide the words into parts according to these segmented points; d) finally order them according to their positions from left to right. The complete process is summarized in Figure 17 in a flowchart. The details are explained below.

1) Let (M) be the left most part of the word. If $(width(M) > 15)$ then go to step 2, else goto to step

ISBN: 978-988-19251-6-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

7.

2) M_{skel} = skeletonized (M).
3) Extract features as explained in sub sub-section II-B1 from M_{skel} and save as Feature Vector (FV).
4) The FV is fed into the trained ANN which gives output(C_{out}) in the form of 26 classes (C) corresponding to each character.
5) Compute $M_{cout} = maximum(C_{out})$.
6) If ($M_{cout} > T$), then it is recognized as a character belonging to class $C(M_{cout})$, where T is user specified recognition accuracy threshold.
7) Else the part right next to M is added to M. If $(width(M) < 50)$ repeat recognition from step 2.
8) This procedure is repeated until a character is recognized with accuracy $(ACU) > T$, or width$(M) > 50$ pixels (character width threshold).

Note: If a character is recognized before $width(M)$ exceeds 50 pixels, then the recognition for the next character would start from the part just right, to the rightmost part of M (character M ends, and process starts with a new character). Else (recognition fails), it would instead start from just next right part of the left most part of M (the left most part is discarded as it does not match). The leftover part of M, which is still to be recognized, is processed again.

The boundary cases of our technique are

1) **Best case:** No overlaps in characters, and $width(M) < 50$ pixels. For example, as shown in Figure 18.

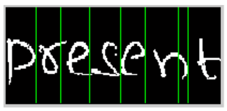


Fig. 18: Best case: detection of good prospective segmentation points.

2) **General case:** when the $15 \leq width(M) \leq 50$ pixels, as we could detect more than required prospective segmentation points between two characters. For example, as shown in Figure 19 and Figure 20.

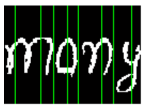


Fig. 19: More than required segmentation points but each character width < 50

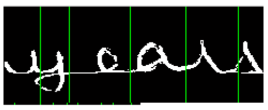


Fig. 20: General case of cursive handwriting

3) **Worst case:** (a) overlapped characters, where perspective segmentation point could not be detected, as shown in Figure 21; (b) $width(M) > 50$ pixels as shown in Figure 22.

For special cases when characters are continuous and cursive, then the system may recognize part of a character as some other character. For example in Figure 22, first two prospective segmentation points of character m may be recognized as character n (false positive).

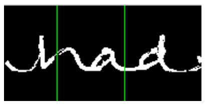


Fig. 21: Worst case (a): no prospective segmentation points between h and a.

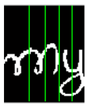


Fig. 22: Worst case (b): $width(M) > 50$ pixel.

III. RESULTS AND DISCUSSION

This paper describes all the necessary steps for offline handwriting recognition system. A new approach for character segmentation and recognition is proposed in the paper. An improvised diagonal based feature extraction technique is used which considerably increases the success rate. In this approach character segmentation and character recognition proceed side by side.

*a) **Preprocessing step:*** Word segmentation (of the proposed system) ACU rate is $> 95\%$ irrespective of cursive or non-cursive handwriting.

*b) **Neural network:*** The ANN with the specified parameters, and the proposed feature extraction technique achieves an ACU of 95.6%. The details of the ANN, for different FV's are enumerated in Table II.

TABLE II: Comparison of Networks.

Networks	1	2	3
Feature Vector (FV)	Vertical	Horizontal	Diagonal, Vertical and Horizontal
Nodes for input	54	54	69
Nodes in hidden layer	30	30	30
Nodes in output layer	26	26	26
Recognition rate	93	92.4	95.6

*c) **Character Segmentation and recognition:*** According to the proposed algorithm and with the help of ANN, the system achieves an ACU of 70% for readable non-cursive handwriting and $50 - 60\%$ in case of cursive handwriting. Compared to POLAR-RADII GRAPHS [14] which have an average ACU of 58%, and Connectionist Character N-grams [1] having average ACU of 41%.

Even in worst cases, when the characters are overlapped and width of some characters exceed 50 pixels our system could recognize cursive and non-cursive handwritings with an average ACU of 55% and 70%, respectively which is significantly better than the other systems.

IV. CONCLUSIONS

With the above results it can be concluded that with more reliable feature extraction technique and larger dataset for character and input sample the result for the complete system can be further improved and enhanced. If the system is trained for a particular handwriting and after that if the system is used to recognize text for that handwriting (used in personalized authentication systems and web logging authentication) it can further improve its success rate to a very appreciable measures. More research on character segmentation may help in making this system a complete offline unconstrained cursive handwriting recognition system. The proposed system achieves an average accuracy of 62.5%.

REFERENCES

[1] Zamora-Martinez, F.; Castro-Bleda, M.J.; Espaa-Boquera, S.; Gorbe-Moya, J.; , "Unconstrained offline handwriting recognition using connectionist character N-grams," Neural Networks (IJCNN), The 2010 International Joint Conference on , vol., no., pp.1-7, 18-23 July 2010

[2] Hong Lee and Brijesh Verma.Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on 1-8 June 2008.page no 2994 - 2999.

[3] Kam-Fai Chan; Dit-Yan Yeung; , "Elastic structural matching for online handwritten alphanumeric character recognition, " Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on , vol.2, no., pp.1508-1511 vol.2, 16-20 Aug 1998

[4] Kumar, M.; Jindal, M.K.; Sharma, R.K.; , "k-nearest neighbor based offline handwritten Gurmukhi character recognition," Image Information Processing (ICIIP), 2011 International Conference on , vol., no., pp.1-4, 3-5 Nov. 2011

[5] Potrus, M.Y.; Ngah, U.K.; Sakim, H.A.M.; AbdulRahman, S.A.; , "Normalization and rectification method for online hindi digit recognition with partial alignment algorithm," Electronics and Information Engineering (ICEIE), 2010 International Conference On , vol.1, no., pp.V1-223-V1-227, 1-3 Aug. 2010

[6] E. Kavallieratou, N. Fakotakis, and G. Kokkinakis. A slant removal algorithm. Pattern Recognition, 33(7):1261-1262, 2000.

[7] Gyeonghwan Kim, Venu Govindaraju, and Sargur N. Srihari. An architecture for handwritten text recognition systems. International Journal on Document Analysis and Recognition, 2(1):37-44, Jul 1999.

[8] U.-V. Marti and H. Bunke. Using a statistical language model to improve the performance of an hmm-based cursive handwriting recognition systems. pages 65-90, 2002.

[9] U.-V. Marti and H. Bunke. Using a statistical language model to improve the performance of an hmm-based cursive handwriting recognition systems pages 65-90, 2002.

[10] Kenneth M. Sayre. Machine recognition of handwritten words: A project report. Pattern Recognition, 5(3):213 - 228, 1973.

[11] Pradeep, J.; Srinivasan, E.; Himavathi, S.; , "Diagonal based feature extraction for handwritten character recognition system using neural network," Electronics Computer Technology (ICECT), 2011 3rd International Conference on , vol.4, no., pp.364-368, 8-10 April 2011

[12] Horita, Y.; Murai, T.; Miyahara, M.; , "Region segmentation using K-mean clustering and genetic algorithms ," Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference , vol.3, no., pp.1016-1020 vol.3, 13-16 Nov 1994

[13] Andria, G.; Savino, M.; Trotta, A.; , "Application of Wigner-Ville distribution to measurements on transient signals," Instrumentation and Measurement, IEEE Transactions on , vol.43, no.2, pp.187-193, Apr 1994

[14] Cajote, R.D.; Guevara, R.C.L.; , "Global word shape processing using polar-radii graphs for offline handwriting recognition," TENCON 2004. 2004 IEEE Region 10 Conference , vol.A, no., pp. 315- 318 Vol. 1, 21-24 Nov. 2004