Online Devanagari Handwriting Word Recognition

A DISSERTATION
submitted towards the fulfillment of the
requirement for the award of the degree of
MASTER OF TECHNOLOGY
in
Computer Science and Engineering

By

PUNEET S POOJARY

Department of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY, ROORKEE
Roorkee - 247667, India

JUNE 2016

AUTHOR'S DECLARATION

I declare that the work presented in this dissertation with title **" Lexicon Free Online Devnagari Word Recognition Using Classifier Combination "** towards the fulfillment of the requirement for the award of the degree of **Master of Technology** in **Computer Science & Engineering** submitted in the **Department of Computer Science & Engineering, Indian Institute of Technology Roorkee, India** is an authentic record of my own work carried out during the period from **June 2015 to May 2016** under the supervision of **Dr. Partha Pratim Roy**, Assistant Professor, Department of Computer Science and Engineering, Indian Institutes of Technology, Roorkee. The content of this dissertation has not been submitted by me for the award of any other degree of this or any other institute.

DATE: ..... SIGNED: .....

PLACE: ..... (PUNEET S POOJARY)

CERTIFICATE

This is to certify that the statement made by the candidate is correct to the best of my knowledge and belief.

DATE: ..... SIGNED: .....

(DR. PARTHA PRATIM ROY)
Assistant Professor
Indian Institutes of Technology, Roorkee

Abstract

Classifier combination is a way to combine the outputs of multiple distinct recognition systems so as to improve the accuracy of the combined system. The training data is used to train multiple instances of neural network based recognition systems (BLSTM). Next the test data is transcribed using each of the trained models. A post processing technique (ROVER) is used to combine the multiple transcriptions obtained for each test sequence into a best transcription. This approach yields a higher transcription accuracy when compared to those obtained with the individual recognition systems used.

Dedication and acknowledgements

Dedicated to my family and friends, for standing by me through thick and thin, without whom I would not have gotten this far.

I would like to express my sincere gratitude to my advisor Dr. Partha Pratim Roy for the continuous support of my study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my study.

I am also grateful to the Dept. of Computer Science, IIT Roorkee for providing valuable resources to aid my research.

TABLE OF CONTENTS

Page

List of Tables **vi**

List of Figures **vii**

1 Introduction **1**
- 1.1 BLSTM . . . 3
- 1.2 HMM . . . 5
- 1.3 ROVER . . . 5

2 Related Works **8**

3 Proposed Model **9**
- 3.1 Preprocessing . . . 10
- 3.2 Feature Extraction . . . 10
  - 3.2.1 Vertical position . . . 12
  - 3.2.2 Curvature . . . 13
  - 3.2.3 Aspect . . . 13
  - 3.2.4 Curliness . . . 14

4 Experimental Result **15**
- 4.1 Dataset . . . 15
- 4.2 Result . . . 15

Bibliography **22**

List of Tables

Table **Page**

4.1 Results for individual models and with rover combination 18
4.2 Transcription results for Test sequence usr81_022t02 19
4.3 Transcription results for Test sequence usr81_025t01 20
4.4 Transcription results for Test sequence usr82_005t01 21

List of Figures

Figure **Page**

1.1 The Devanagari alphabet 2
1.2 Possible positions for placing matras on a base consonant in the Devanagari script [1]. 3
1.3 (a) Online handwriting data. (b) Offline handwriting data 4
1.4 A Recurrent Neural Network. 4
1.5 Long Short Term Memory 6
1.6 Bidirectional Recurrent Neural Network 6
1.7 Rover system architecture 7

3.1 Training multiple sequence recognizers using the train data set. 10
3.2 Combining the outputs of multiple models to obtain best transcription. 11
3.3 NPen++ writing direction 12
3.4 NPen++ curvature 13

4.1 Variation in writing styles for data sequence 001t01 16
4.2 Variation in writing styles for data sequence 007t01 17
4.3 Test sequence usr81-022. 19
4.4 Test sequence usr81-025. 20
4.5 Test sequence usr82-005. 21

CHAPTER 1

INTRODUCTION

Methods for recognizing handwritten text have been an active area of interest for several years. This enduring interest in handwriting recognition is proof of the diverse applications for such systems. An accurate handwriting recognition system can provide a natural interface to computing systems that otherwise require users to master input devices for their operation. This can help a majority of the technically illiterate population gain access to computers which can benefit them immensely.

Devanagari is an alphabet widely used in many parts of India and Nepal.Characters are written left to right along with the distinctive horizontal line known as shirorekha. The Devanagari alphabet has 14 vowels and 33 consonants. Hindi which uses the Devanagari script is the worlds third most commonly used language after English and Chinese. Approximately 500 million people read and write hindi. The typical Devanagari alphabet is shown in Figure 1.1.

The Devanagari script has some unique complications when compared to other scripts like latin. This makes the process of adapting recognition systems built for other scripts into Devanagari quite difficult. For example even though consonants are written linearly in a left to right order, matras which modify the base consonant can be placed non-linearly above, below or on either side of the consonants. An example of this is shown in Figure1.2

Recognition systems that fail to account for this disparity in spatial and temporal sequencing often perform poorly while processing Devanagari text.

Handwriting recognition systems can be broadly classified into offline and online,

Vowels *(swara)*	अ	अं	आ	आं	इ
	इ	ई	उ़	उ	उं
	ऊ	ऊं	ऋ	ॠ	ऌ
	ऌ	ऐ	ए	ऍ	ऐ
	ओ	ओ	ऑ	औ	अं
	अः				
Vowel Marks *(maatra)*	◌	◌ा	◌ा	◌ि	◌ि
	◌ी	◌ु	◌ु	◌ु	◌ू
	◌ृ	◌ॄ	◌ॢ	◌ॣ	◌ॅ
	◌े	◌ॅ	◌ै	◌ौ	◌ं
	◌ः	◌ऽ			
Consonants *(vyanjana)*	क	ख	ग	घ	ङ
	च	छ	ज	झ	ञ
	ट	ठ	ड	ढ	ण
	त	थ	द	ध	न
	प	फ	ब	भ	म
	य	र	ल	व	श
	ष	स	ह	ळ	

Figure 1.1: The Devanagari alphabet .

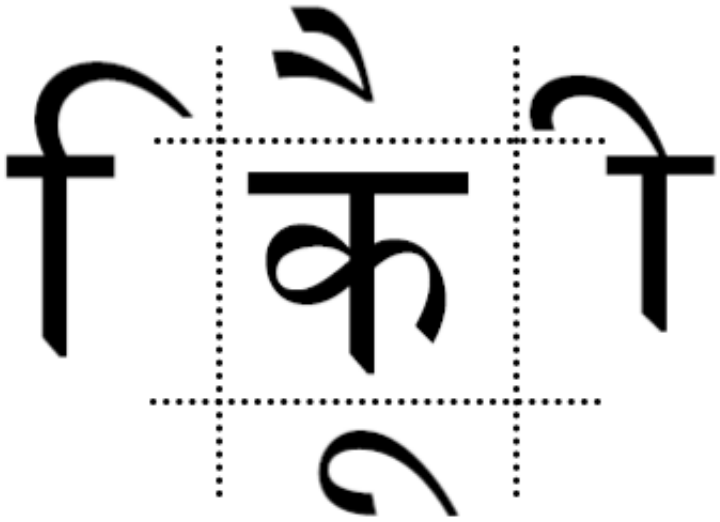


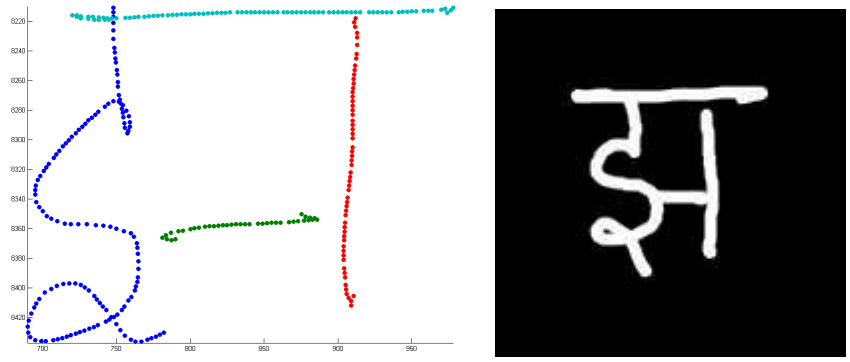
Figure 1.2: Possible positions for placing matras on a base consonant in the Devanagari script [1].

depending on the nature of the data they can process. Offline handwriting is used to recognize text in static images. This is mainly used to transcribe scanned documents. Online handwriting recognition systems work with a recording of the pen tip as the characters are written. This is typically represented as a sequence of pen tip positions, ordered in time. This temporal information that is provided in online data can be used to give additional information about the written text, which is absent in offline data. Online handwriting recognition systems seek to exploit this temporal information to increase the accuracy of recognition. Online recognition systems can be used to provide natural interfaces to computers which can help the large number of digitally challenged people obtain access to computers.

1.1 BLSTM

The basic architecture of recurrent neural networks consists of a network of units called neurons or nodes, each of which can have directed connections to the other nodes in the network. The nodes can be divided into input, output and hidden nodes. The incoming connections to each node have distinct modifiable weights. The node outputs a real value using an activation function on the input values. Figure 1.4 shows how the cells in RNN‚Äôs can have backward and self connections.

Recurrent Neural networks (RNN) have shown good results in sequence learning



(a) (b)

Figure 1.3: (a) Online handwriting data. (b) Offline handwriting data .

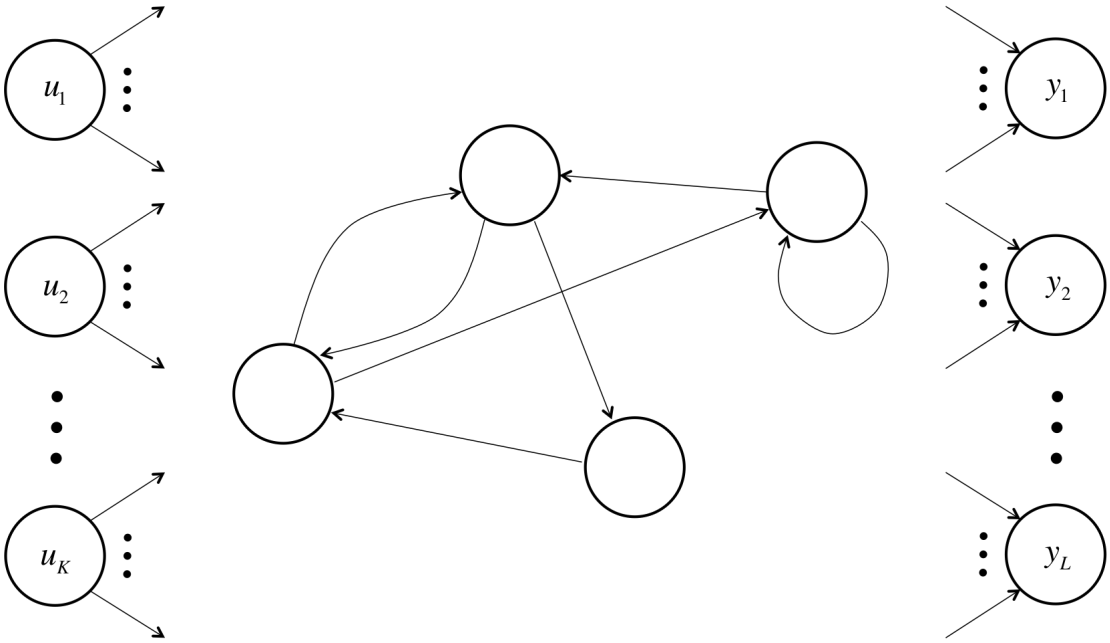


Figure 1.4: A Recurrent Neural Network.

and recognition tasks. This can be attributed to the ability of RNN's to consider previous state information also known as memory. This can provide a useful context for processing the current inputs. However RNN's suffer from limitations to the range of context that can be accessed. This is because under some conditions, the influence of a particular input on the hidden layer as well as the network layer is prone to blow up or decay exponentially, as it is cycled through recurrent connections. This is often referred to as the vanishing gradient problem.

In order to overcome this problem Long Short Term Memory (LSTM) was introduced [2]. LSTM relies on gate units to control the input, output and memory in each cell. This gives the net control over how the error is propagated through the recurrent connections. As a result LSTM can store and access information over a much longer period of time when compared to normal RNN's, by overcoming the vanishing gradient problem. The architecture of a single LSTM cell is shown in Figure 1.5.

Another problem with RNN's while processing sequences is that they have access to past information, but do not have any idea of the future context. Bidirectional Long Short Term Memory (BLSTM) is an architecture designed to overcome this problem Here the input is provided to two separate layers, which process the input in different directions, as shown in Figure 1.6. Both of these layers are eventually connected to the output unit, which enables the network to have information about the input in both directions [3].

1.2 HMM

A Hidden Markov Model (HMM) is a doubly stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observed symbols [4].

Hidden Markov Models have been used for a long time as part of audio and sequence recognition problems due to their performance in temporal pattern recognition problems.

Forced Viterbi Alignment is a method of matching a given label to a part of an input sequence. The algorithm returns the most probable alignment of a sequence of labels in the test sequence.

1.3 ROVER

Recognition Output Voting Error Reduction (ROVER) is originally an algorithm intended to improve the recognition accuracy for automated speech recognition systems [5]. It

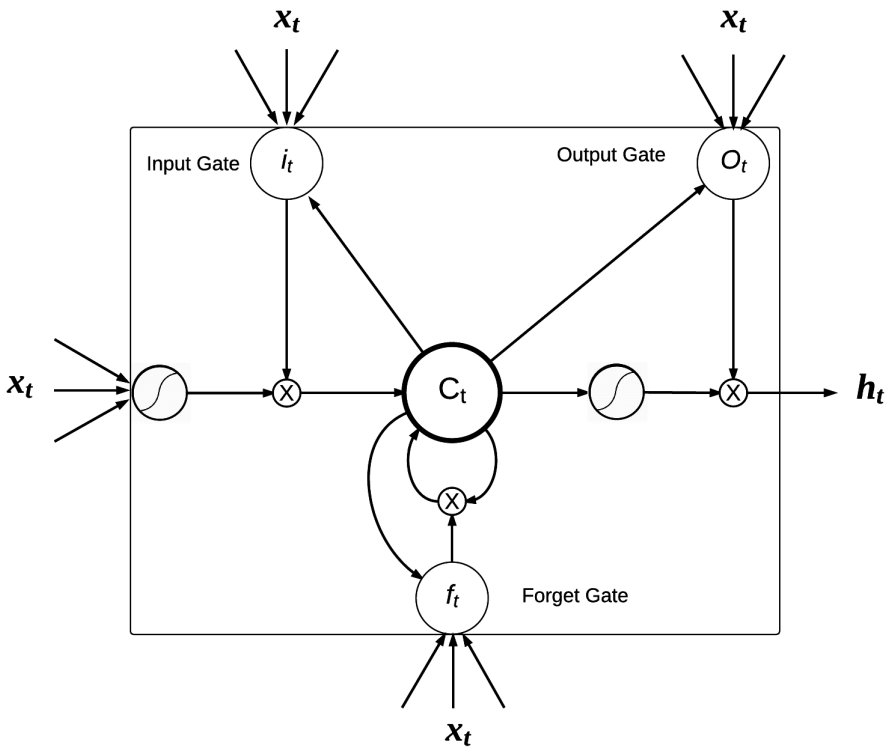


Figure 1.5: Long Short Term Memory

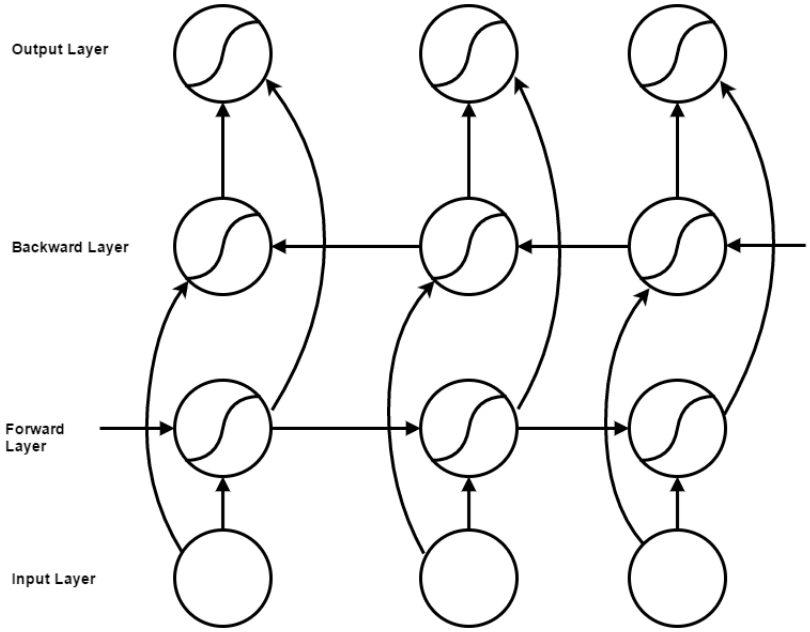


Figure 1.6: Bidirectional Recurrent Neural Network

works by exploiting differences present in the type of errors made by different recognition systems.

The rover system works in two modules. In the first module we take the input of multiple hypotheses from different recognition systems. The hypotheses for each sequence are then aligned for the next step. Figure 1.7 shows the basic rover system architecture.

Once the aligned sequences are obtained the voting module processes it to obtain the best transcription sequence.

Since the rover module is independent of the recognition system, it can be used to combine a wide variety of individual recognizers, with minor changes to the output formats. This can be useful for processing recognition systems for multiple languages using the same module.

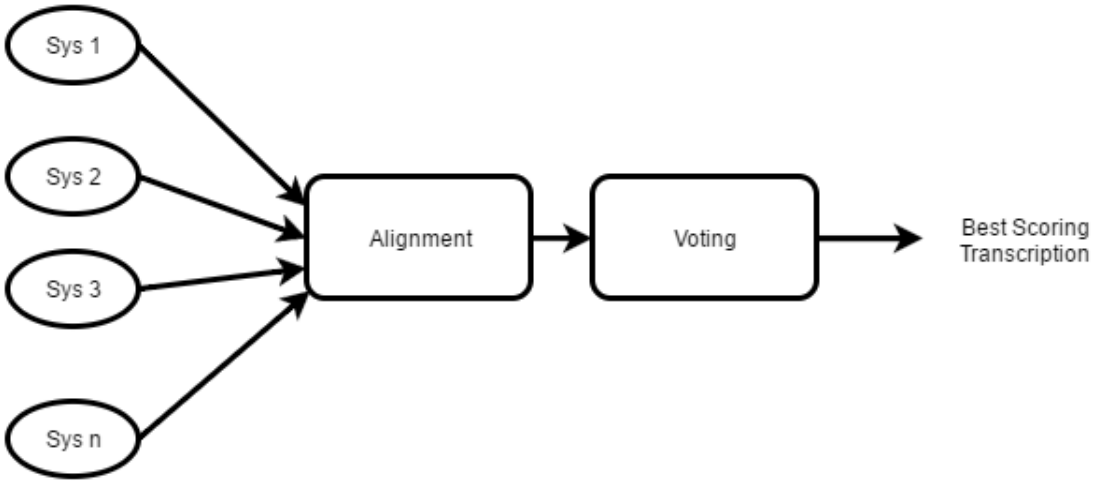


Figure 1.7: Rover system architecture

CHAPTER 2

RELATED WORKS

Prior work on lexicon free Devanagari online text recognition is very limited. Swethalakshmi [6] proposed a recognition system based on explicit grouping of strokes based on "proximity analysis" into characters, and recognizing each stroke within. Any problems involving oversegmentation or undersegmentation is resolved by postprocessing methods which use the stroke labels to classify the characters. The best word is evaluated by interpreting individual characters and using a lexicon based approach or similar language based models. Manual analysis of the training data is required to determine unique strokes and the different writing styles of a character.

Identifying words by recognizing the individual stroke segments in a similar way has also been proposed for other Indian languages such as Bangla [7] and Gurmukhi [8]. One of the major issues with these approaches is that they are not easy to scale for recognizing other Indic scripts since they rely on script dependent features and require manual modifications for training.

Hidden Markov Models are a popular choice for online handwriting recognition systems such as Latin and CJK scripts [9], [10] because of their capacity to recognize sequential data. For recognizing Latin script, we need to segment the cursive writing into its component characters, HMM's are used to identify at the character level and are then combined to form word HMM's. Word HMM's are then used to segment the input data into letters alongwith recognition. CJK recognition follows a different approach, with HMM's being build for the stroke level recognition and then the models are connected to form a large network that is used to represent different stroke orders.

CHAPTER 3

PROPOSED MODEL

We attempt to improve the accuracy of online devanagari word recognition by combining the outputs of multiple classifiers so as to obtain the best output sequence. Neural Networks are initialized with random values, so as a result after training each model is inherently different from any other instance of a trained model using the same training data. This difference can be seen by comparing the results obtained by different models using the same training sequence. The variation in trained neural network models manifests as discrepancies in the ability to identify some sequences correctly. If we can correctly recognize the sequences that are correctly identified by a particular neural network, and incorrectly identified by another network, we should be able to improve the transcription result accuracy with the help of multiple models. Before we can do this we need to have several good transcription models, a method to align different transcription results and finally a way to merge results optimally. Here we use BLSTM for transcription, HMM for alignment and ROVER for combining results.

In the first step shown in Figure 3.1 the training data is used to train multiple neural networks based on BLSTM. This gives us multiple models for the next step. Simultaneously we use the training data to extract features and train an HMM for character level recognition using the training data.

Next using the n models from the first step we recognize the test data, to get n output sequences. These n outputs are aligned using the HMM models trained previously, which adds alignment information to the output data. The n sequences are then processed using the rover algorithm to get best output sequences for each test data. This process is

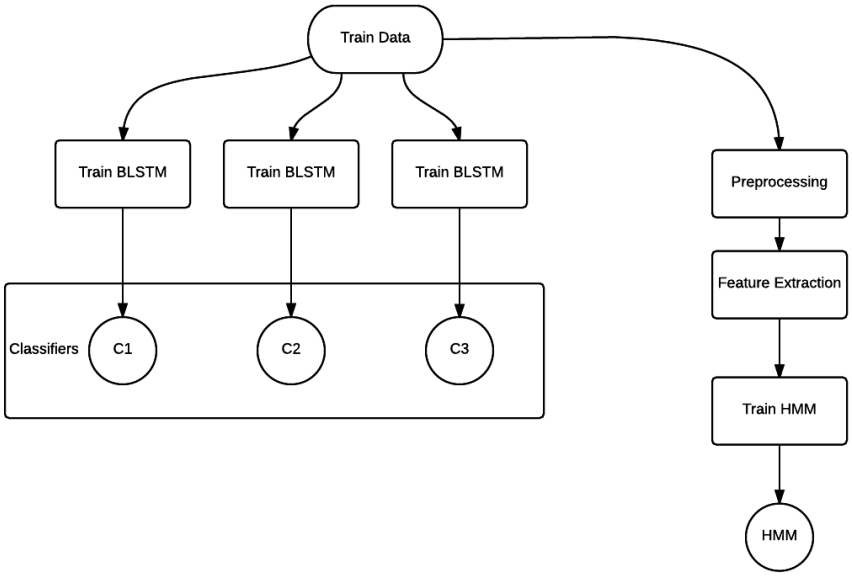


Figure 3.1: Training multiple sequence recognizers using the train data set.

shown in Figure 3.2.

For training with BLSTM, the data was split 70:30 into training and testing sets. The network configuration used was three hidden layers with 20, 60 and 180 nodes each. Five different networks were trained with the available training data. These five models were then used to transcribe the test data sequences. Consequently each test sequence would have five different transcriptions provided by the five models.

3.1 Preprocessing

The data which is provided in the unipen format is first resampled using interpolation to ensure uniform distance between consecutive points. Next the data is scaled to the 0 to 1 range, and invalid coordinates are discarded.

3.2 Feature Extraction

There exist very few feature sets which are compatible with online data formats as compared to offline data, where a large number of feature descriptors are easily available. For extracting features from the online dataset, we chose the npen++ feature set. The

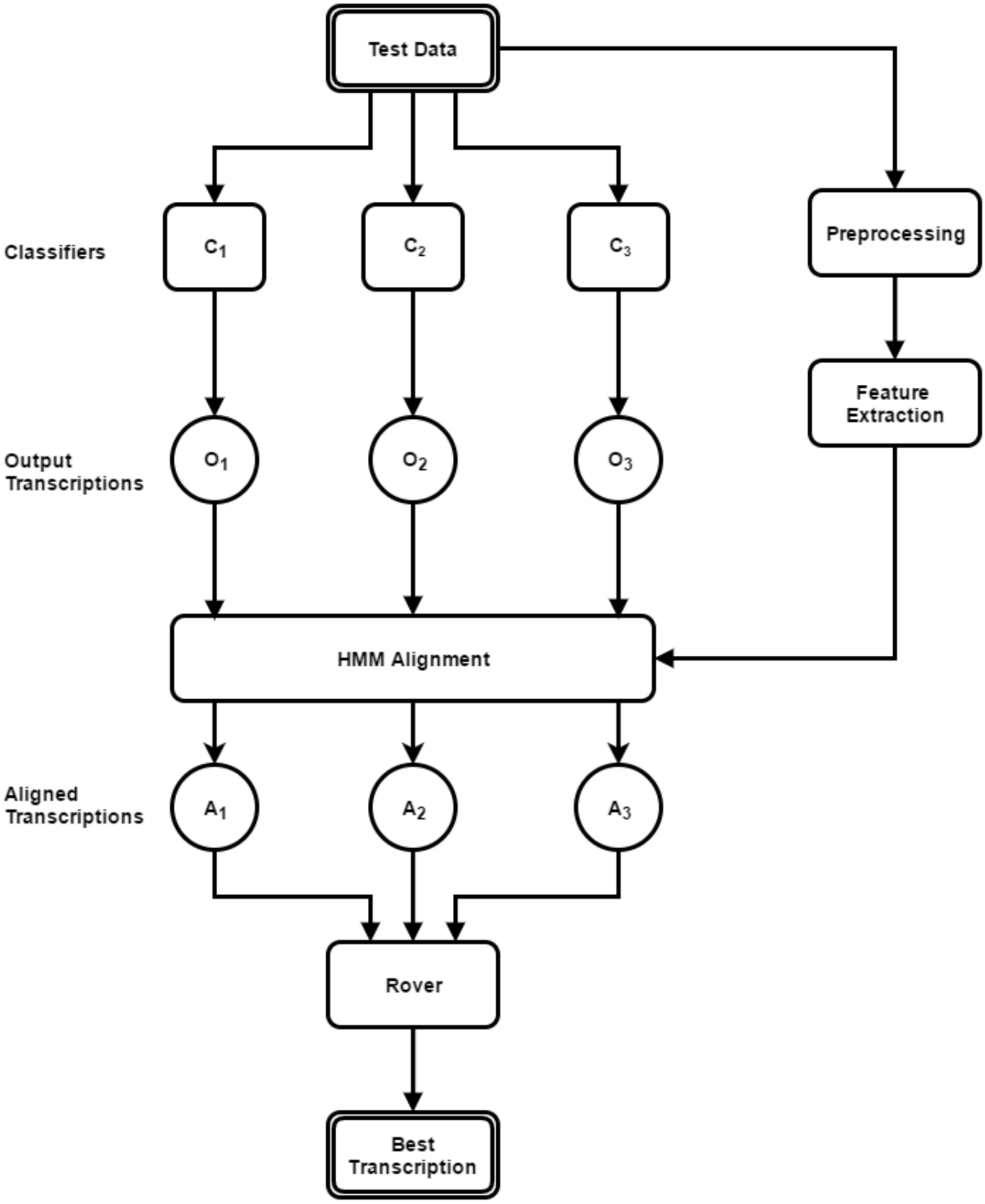


Figure 3.2: Combining the outputs of multiple models to obtain best transcription.

NPen++ features require a normalized sequence of points (x(t),y(t)), which are then used to calculate the feature sequence. The features used are as follows.

3.2.1 Vertical position

The vertical position for a point(x(t),y(t)) is the vertical distance of the point from the x axis, or the baseline for the sequence. It can be considered negative if the point is below the baseline.

$$V(t) = y(t) - b(y(t)) \tag{3.1}$$

3.2.1.1 Writing direction

This feature describes the writing direction in a local context. It gives the cosine and sine components for the writing direction.

$$cos\alpha(i) = \frac{\Delta x(i)}{\Delta s(i)} \tag{3.2}$$

$$sin\alpha(i) = \frac{\Delta y(i)}{\Delta s(i)} \tag{3.3}$$

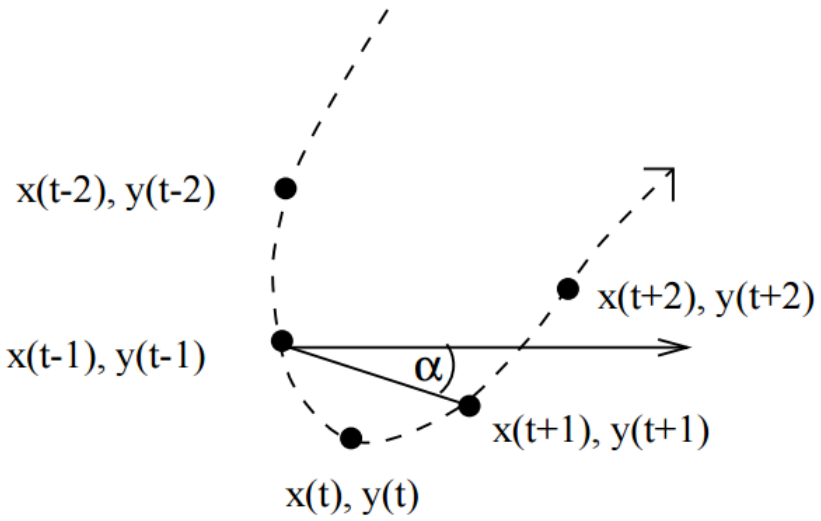


Figure 3.3: NPen++ writing direction

where $\Delta x, \Delta y$ and Δs are computed as follows:

$$\Delta s(i) = \sqrt{\Delta x^2 + \Delta y^2} \tag{3.4}$$

$$\Delta x(i) = x(i-1) - x(i+1) \tag{3.5}$$

$$\Delta y(i) = y(i-1) - y(i+1) \tag{3.6}$$

3.2.2 Curvature

The computation of curvature at a point $(x(i), y(i))$ consider the previous and next point to that point and is describe as follows:

$$cos\beta(i) = cos\alpha(i-1) * cos\alpha(i+1) + sin\alpha(i-1) * sin\alpha(i+1) \tag{3.7}$$

$$sin\beta(i) = cos\alpha(i-1) * sin\alpha(i+1) + sin\alpha(i-1) * cos\alpha(i+1) \tag{3.8}$$

Note that this sequence does not actually compute curvature but compute angular

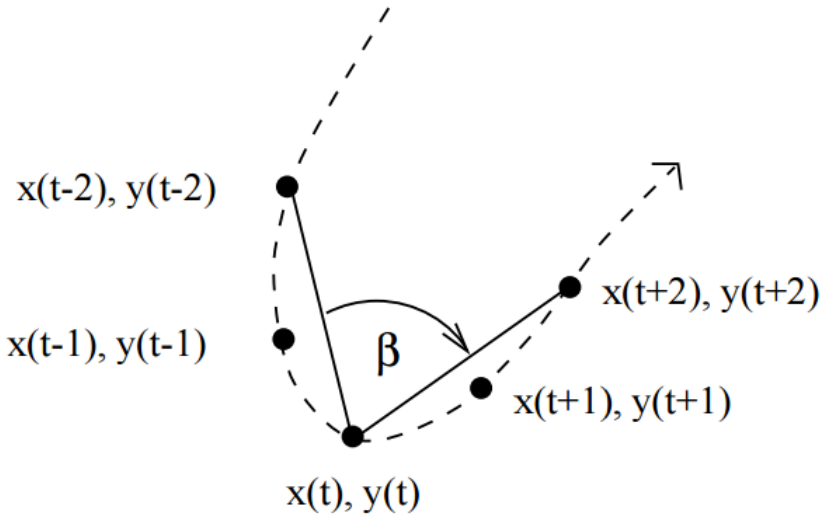


Figure 3.4: NPen++ curvature

difference which suffice in our case.

3.2.3 Aspect

The ratio of the height to the width of the bounding box accommodating the succeeding and preceding points of $(x(i), y(i))$ is the aspect $A(i)$ of the contour at point i. It is computed as:

$$A(i) = \frac{\Delta y(i) - \Delta x(i)}{\Delta y(i) + \Delta x(i)} \tag{3.9}$$

3.2.4 Curliness

The deviation from a straight line in the neighborhood of $(x(i), y(i))$ describe Curliness $C(i)$ feature. It is computed as the ratio of length of the contour and larger side of the bounding box.

$$C(i) = \frac{L(i)}{max(\Delta x, \Delta y)} - 2 \tag{3.10}$$

where $L(i)$ is the length of contour in the neighborhood of the point computed as the sum of all line segments in the neighborhood of the point. Δx and Δy are width and height of the bounding box.

CHAPTER 4

EXPERIMENTAL RESULT

4.1 Dataset

The dataset used is the Isolated Handwritten Devanagari Word Dataset provided by HP Labs India. The dataset has 220 word samples of 70 hindi words recorded by 110 native writers. The 70 words were selected to represent commonly used hindi words, and to cover all symbols in the Devanagri script. The data was recorded by using Acecad Digimemo A402 graphic tablet, and stored in standard unipen format.

4.2 Result

For testing the proposed model, the data was split 70:30 into training and testing sets. The training set was used to train 5 different BLSTM instances, each of which provided two different models for best label error and best ctc error. Simultaneously a HMM model was trained using the training set. The test set was processed using each of these 10 models to get 10 different transcriptions. The transcriptions were aligned using Viterbi forced-alignment with the trained HMM model. Next the aligned transcriptions for each sequence were combined using ROVER to obtain a best transcription for each test sequence. Table 4.1 shows the results obtained using individual classifiers to recognize the test set along with the result obtained after using ROVER.

The benifit obtained by using ROVER can be seen in the analysis of results for individual test sequences shown in Tables 4.2 , 4.3 and 4.4. For test sequence usr81_022t02 the

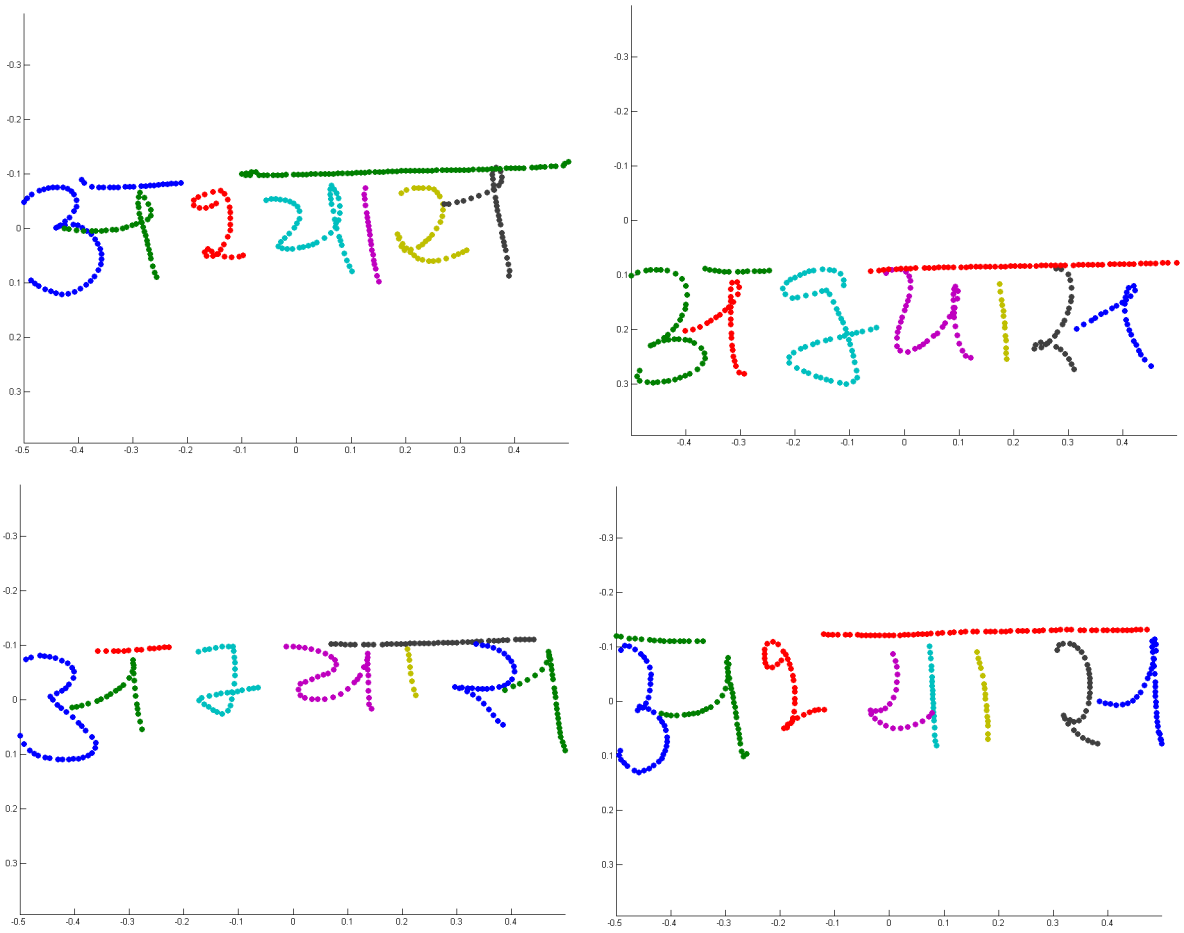


Figure 4.1: Variation in writing styles for data sequence 001t01

incorrect predictions by classifiers C4 and C1 are correct in the final output by ROVER.

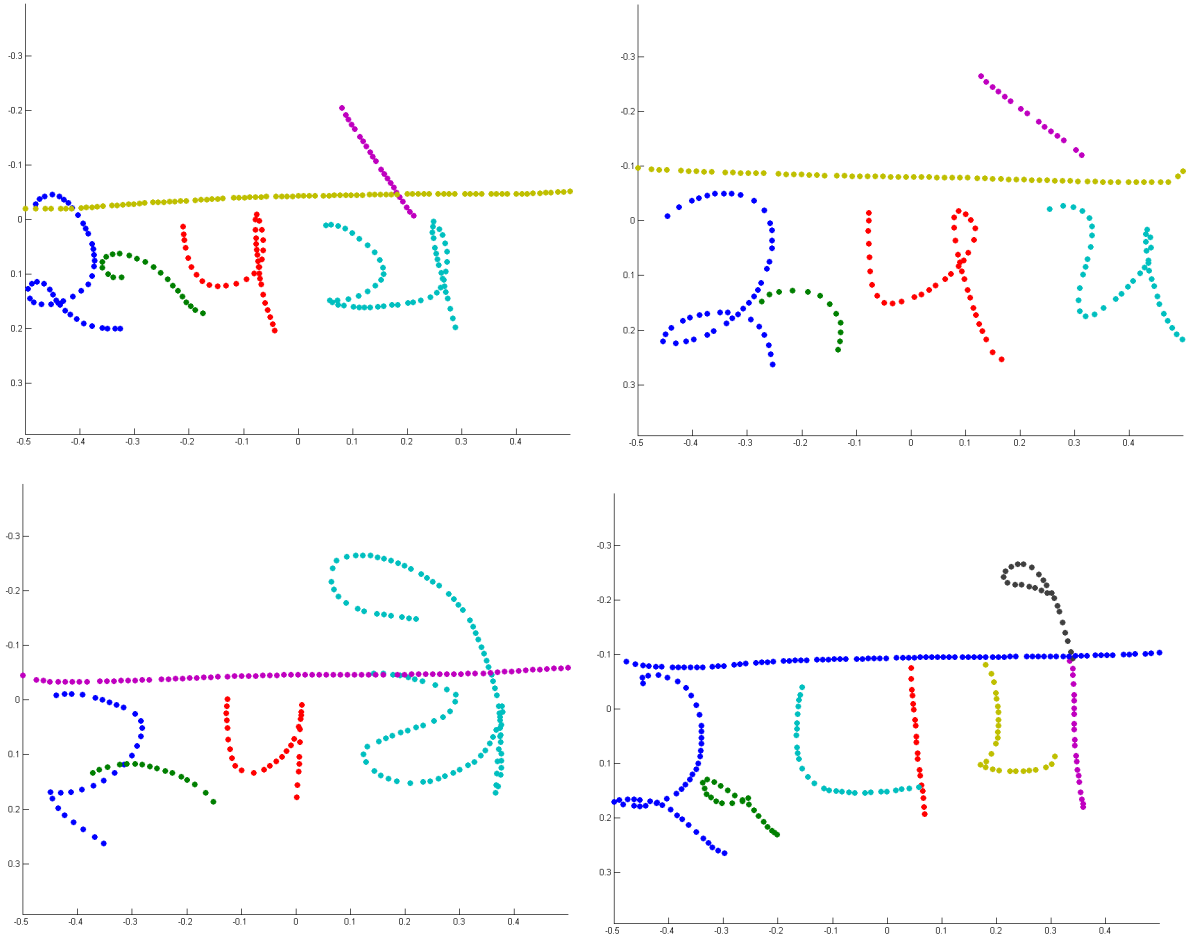


Figure 4.2: Variation in writing styles for data sequence 007t01

Table 4.1: Results for individual models and with rover combination

Classifier	**Accuracy**
C 1	82.40
C 2	82.40
C 3	82.45
C 4	82.45
C 5	81.70
C 6	81.70
C 7	82.35
C 8	82.51
C 9	83.22
C 10	83.22
Rover	**88.83**

Table 4.2: Transcription results for Test sequence usr81_022t02

Classifier	**Transcription**						
Ground Truth	34	51	24	62	25	48	0
Rover	34	51	24	62	25	48	0
C1	34	**63**	**40**	62	25	**34**	0
C2	34	51	24	62	25	48	0
C3	34	51	24	62	25	48	0
C4	**66**	**–**	24	**40**	25	**34**	0
C5	34	51	24	62	25	48	0

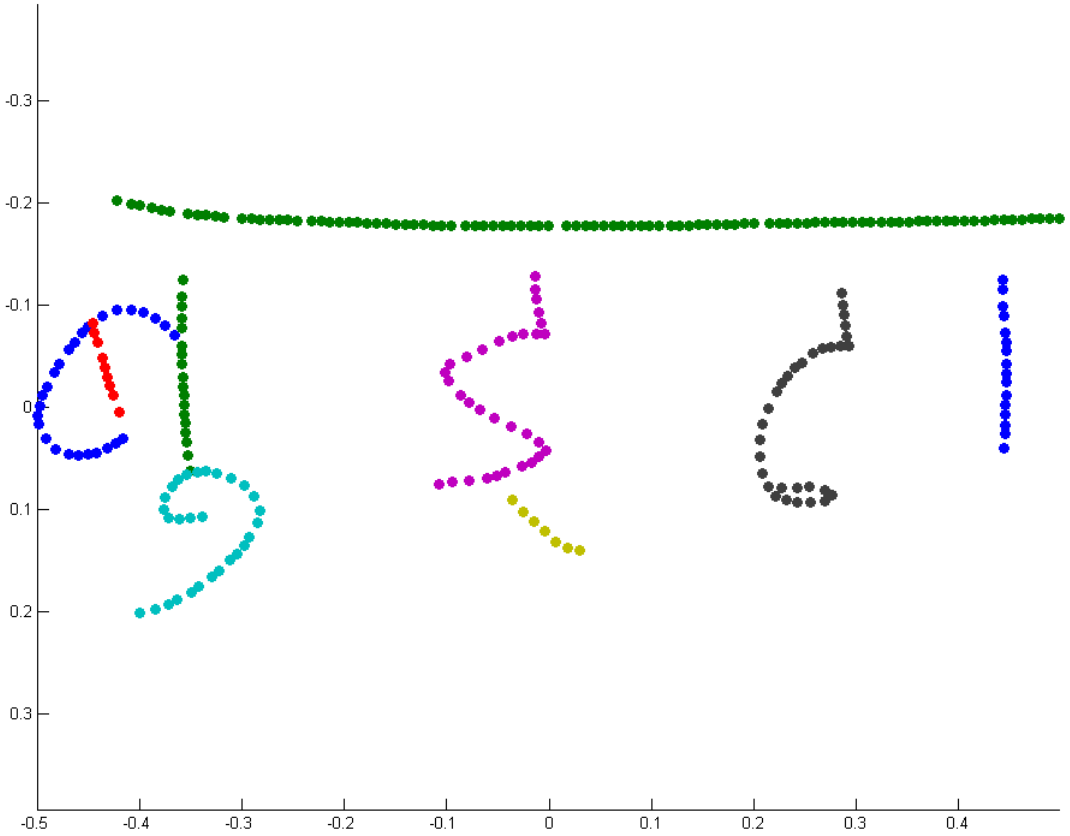


Figure 4.3: Test sequence usr81-022.

Table 4.3: Transcription results for Test sequence usr81_025t01

Classifier	**Transcription**					
Ground Truth	104	50	36	27	50	0
Rover	104_50	50	36	43	50	0
C1	104_50	54	58	43	50	0
C2	104_50	50	36	27	50	0
C3	104	50	36	43	50	0
C4	104	50	36	27	50	0
C5	104_50	–	36	0_27	–	–

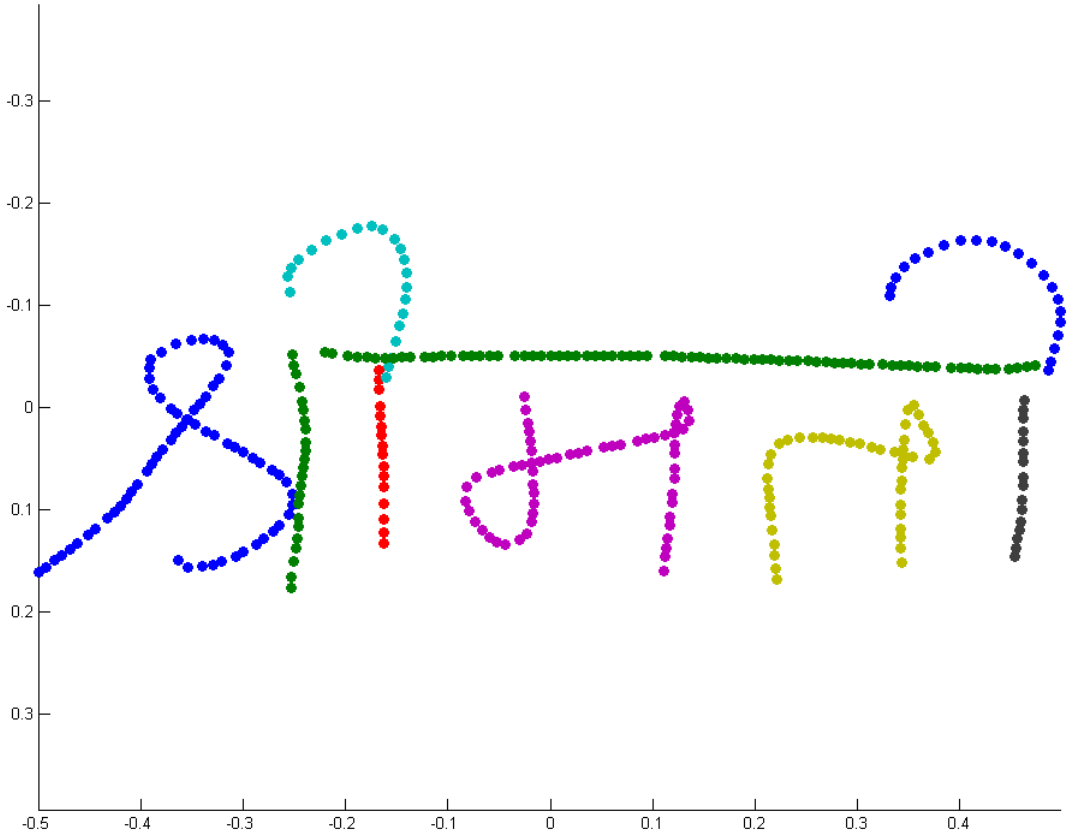


Figure 4.4: Test sequence usr81-025.

Table 4.4: Transcription results for Test sequence usr82_005t01

Classifier	**Transcription**					
Ground Truth	38	109	32	37	54	0
Rover	38	109	32	37	54	0
C1	77	–	–	37	31	0
C2	38	109	32	37	54	0
C3	77	109	32	37	54	0
C4	38	109	32	37	54	0
C5	38	109	32	37	–	0

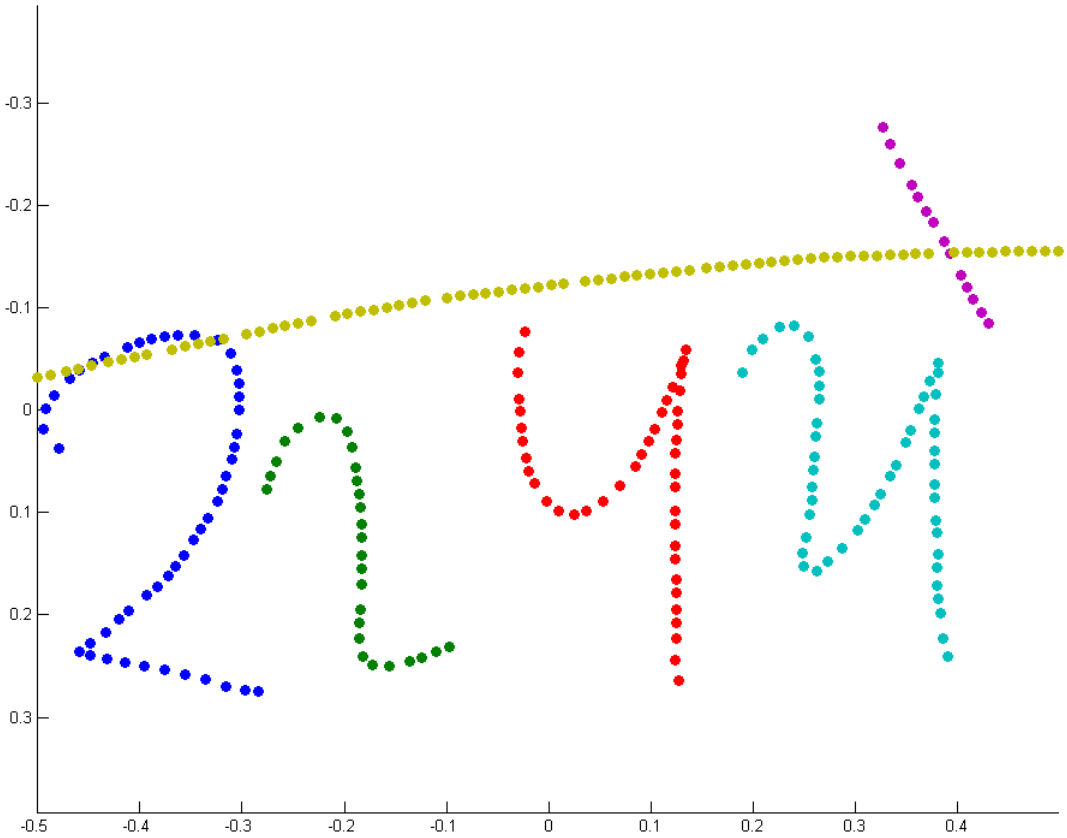


Figure 4.5: Test sequence usr82-005.

BIBLIOGRAPHY

[1] V. Govindaraju and S. Setlur, *Guide to OCR for Indic Scripts*, Springer, 2009.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation* **9**(8), pp. 1735–1780, 1997.

[3] A. Graves, *Supervised sequence labelling*, Springer, 2012.

[4] L. R. Rabiner and B.-H. Juang, "An introduction to hidden markov models," *ASSP Magazine, IEEE* **3**(1), pp. 4–16, 1986.

[5] J. G. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover)," in *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, pp. 347–354, IEEE, 1997.

[6] H. Swethalakshmi, A. Jayaraman, V. S. Chakravarthy, and C. C. Sekhar, "Online handwritten character recognition of devanagari and telugu characters using support vector machines," in *Tenth International workshop on Frontiers in handwriting recognition*, Suvisoft, 2006.

[7] U. Bhattacharya, A. Nigam, Y. Rawat, and S. Parui, "An analytic scheme for online handwritten bangla cursive word recognition," *Proc. of the 11th ICFHR* , pp. 320–325, 2008.

[8] A. Sharma, R. Kumar, and R. Sharma, "Rearrangement of recognized strokes in online handwritten gurmukhi words recognition," in *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pp. 1241–1245, IEEE, 2009.

[9] J. Hu, S. G. Lim, and M. K. Brown, "Writer independent on-line handwriting recognition using an hmm approach," *Pattern Recognition* **33**(1), pp. 133–147, 2000.

[10] M. Nakai, N. Akira, H. Shimodaira, and S. Sagayama, "Substroke approach to hmm-based on-line kanji handwriting recognition," in *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pp. 491–495, IEEE, 2001.