# Domain based Bilingual Handwriting Recognition for Gurumukhi-English Script

*A **Thesis***

*submitted in partial fulfillment of the requirements for the award of degree of*

***Doctor of Philosophy***

*by*

**Sukhandeep Kaur**

(951603011)

under the guidance of

**Dr. Seema Bawa**

**Professor, Computer Science and Engineering Department,**

**TIET, Patiala-147004, INDIA**

**Dr. Ravinder Kumar**

**Associate Professor, Computer Science and Engineering Department,**

**TIET, Patiala-147004, INDIA**

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

**Computer Science and Engineering Department**

**Thapar Institute of Engineering and Technology, Patiala -147004, INDIA**

**May 2024**

# Certificate

I, Sukhandeep Kaur, Regn No. 951603011, hereby declare that the work which is being presented in this thesis entitled, "Domain based Bilingual Handwriting Recognition for Gurumukhi-English Script" in partial fulfillment of the requirement for the award of "Doctor of Philosophy" submitted in Computer Science and Engineering Department of TIET, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Seema Bawa and Dr. Ravinder Kumar, refers other research works which have been duly listed in the reference section. The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

(Sukhandeep Kaur)

Regn No. 951603011

*This is to certify that the above statements made by the candidate is correct and true to the best of my knowledge.*

Verified by:

(Dr. Seema Bawa)

Computer Science and Engineering Department, TIET Patiala-147001, India

(Dr. Ravinder Kumar)

Computer Science and Engineering Department, TIET Patiala-147001, India

# Acknowledgement

First and foremost, I offer my profound gratitude to the Almighty, whose grace and blessings have guided me through every step of this challenging yet rewarding journey. As Gurbani lines proclaim, “The Lord himself has stood up to resolve the affairs of the Saints; He has come to complete their tasks”.

In my third grade, I queried my grandfather, Sardar Darshan Singh, about the highest degree. He imparted that education knows no bounds, but a Ph.D. bestows the title of Dr. Since then, obtaining this distinction became a cherished aspiration for me. I extend my gratitude to all my Gurus and mentors who have guided me from my earliest school days to the culmination of my PhD research.

I owe my sincere gratitude to my supervisors Dr. Seema Bawa and Dr. Ravinder Kumar for their essential guidance and support at every stage of my research work. Without their unwavering confidence in me and support, this thesis would not have been possible. I would like to thank Dr. Seema Bawa, who suggested me this research topic and introduced me to the world of Artificial Intelligence. Without her continuous optimism concerning this work, encouragement and blessings, this research would hardly have been completed. I am eternally grateful to my supervisor for instilling discipline in me through her exemplary conduct and leadership. I also express my warmest gratitude to my other supervisor, Dr. Ravinder Kumar, for introducing me the interesting world of computer vision and text recognition for regional languages. His guidance and supervision have been a true support during this work. Beyond my supervisors responsibilities as academic supervisors, their assistance on a human level has been crucial to the success of this thesis and would not have been possible without them. And for this I am truly grateful. They are great mentors for my life as well.

I am thoroughly grateful and highly indebted to Dr. N.Tejo Prakash (Dean of Research)

for his constant support. I am grateful to Dr. Shalini Batra (Head of CSE department), Dr. Jhillik Bhattacharya (Associate Head, CSE department), Dr. Ajay Kumar (Associate Head, CSE department) for providing me with the best research environment in the department. I also acknowledge the Council of Scientific and Industrial Research (CSIR), India for funding this research project. Additionally, I appreciate the guidance and suggestions provided by the doctoral committee members, including Dr. RK Sharma, Dr. Ajay Kumar, and Dr. MD Singh.

I am immensely grateful to my family for their unwavering love, encouragement, and sacrifices throughout my doctoral journey. To my parents, Mr. Jagtar Singh and mother Mrs. Satwinder Kaur your endless support and belief in me have been the driving force behind my success. Your sacrifices and dedication to my education have laid the foundation for this achievement, and for that, I am eternally grateful.

To my dear in-laws, Mr. Amrik Singh and my mother-in-law Mrs. Jasvir Kaur thank you for your understanding, encouragement, and constant support throughout this demanding period. Your belief in my abilities and your unwavering support have been a source of strength and motivation. . I do not think I can ever repay the dept I owe them.

I owe a debt of gratitude to my spouse, Mr. Gaganpreet Singh Tiwana for their unwavering love, patience, and understanding during the ups and downs of my doctoral journey. Your support and encouragement have been my rock, and I am deeply thankful for your presence by my side every step of the way. Words can never be enough to thank his constant encouragement.

I would also like to thank my friends Mrs Rashmeet Toor, Mr. Sachendra, Miss. Akashdeep Kaur, Miss Harshita, Miss Sunandini, Mrs Ravneet Sodhi and Mr. Ajay Kumar for helping and encouraging me.

## Abstract

Handwriting is the most popular and essential way of written communication in day-to-day life. Even in this digital age, handwritten documents are still very much used in public offices to maintain the records. People feel more comfortable in handwriting as compared to typing the text. Thus, there arises the need of some handwritten text recognition system for the digitization of documents. Due to the presence of large number of writing systems and scripts across the world with variety of character sets, it is very difficult to develop a single text recognition system for all the scripts. Many researches have explored machine learning and traditional feature extraction approaches for multilingual text recognition in handwritten and printed text.

An efficient text recognition system should recognize the text with significant variations in individual handwriting styles despite the age, gender, and educational background of the writers. Many scripts and languages have similar shaped character set which acts as major challenge for multilingual text recognition. Every domain has different category of users and writing content. Hence, to improve the performance of text recognition, it is needed to develop a text recognition system corresponding to each domain which limits the targeted users and content for training data.

Multilingual text recognition has various phases like data acquisition, pre-processing, segmentation, script identification and character recognition etc. Numerous research attempts have been made for improving the performance of each phase of text recognition using traditional, deep learning, machine learning, transfer learning and ensemble machine learning approaches. Due to shortage of benchmark datasets, deep learning based techniques are less explored for regional languages. In this thesis domain based bilingual handwritten text recognition system for Gurumukhi-English script is proposed. In the proposed text recognition system i.e. Bilingual Handwritten Text Recognition for Academic Domain (BHTRforAD), two new datasets, three segmentation approaches and three OCRs are introduced. Bilingual dataset from Academic domain containing text written in Gurumukhi and English script with large variations in content, writing style and document style has been designed. Further, for

composite character recognition of Gurumukhi script, composite character dataset with 307 classes is designed. To segment the handwritten documents, three segmentation approaches corresponding to line, word and character are proposed using heuristic approaches. The proposed segmentation approaches are able to segment the text with curved, skewed, closed and touching text lines and words with inter and intra word gap efficiently. Further, to evaluate the deep learning and traditional feature extraction methods for text recognition, this work considers three traditional methods (GLCM, HOG, Gabor) and three deep learning based approaches (LeNet, Vgg19, ResNet50). Similarly, for classification, many machine learning based approaches like SVM, RF, KNN etc. along with ensemble machine learning approaches are considered. For script identification, various combinations of traditional and deep learning based features and classifiers are evaluated to find the best combination of feature set and classifier. In OCR, three OCRs are designed corresponding to scripts i.e. Gurumukhi OCR, English OCR and alphanumeric OCR. For Gurumukhi script, a segmentation free approach to recognize the composite characters for Gurumukhi is proposed using two stage classification approach.

To check the performance of proposed algorithms various performance measures are used like accuracy, precision, recall, F1 score, detection rate, recognition accuracy, and CPU time etc. The results obtained from the experiments have proved the efficiency of proposed algorithms for text recognition of bilingual handwritten documents. Finally, a case study considering Academic domain documents has been conducted to test the proposed system. Number of test cases have been designed corresponding to each phase of bilingual text recognition. Documents containing text with large variations are considered as input for text recognition. Test cases designed for line, word and character segmentation are successfully passed for skewed, straight and curved text with inter and intra word gap. Similarly, test cases for script identification and OCRs have been conducted and results are analyzed.

# List of Publications

1. Kaur, Sukhandeep, Seema Bawa, and Ravinder Kumar. "A survey of mono-and multilingual character recognition using deep and shallow architectures: indic and non-indic scripts." Artificial Intelligence Review 53 (2020): 1813-1872.(SCI IF=12)

2. Kaur, Sukhandeep, Seema Bawa, and Ravinder Kumar. "Segmentation-free composite character recognition (CR) in bilingual handwritten text for Gurumukhi–English scripts." Soft Computing (2023): 1-20.(SCI IF=4)

3. Kaur, Sukhandeep, Seema Bawa, and Ravinder Kumar, "Heuristic-based Text Segmentation of Bilingual Handwritten Documents for Gurumukhi-Latin Scripts." Multimedia Tools and Applications. (Accepted) (IF=3.2).

4. Kaur, Sukhandeep, Seema Bawa, and Ravinder Kumar, "Bilingual Documents Text line extraction using conditional GAN" has been published in **ICMLBDA 2022 at IIT Patna**.

5. Kaur, Sukhandeep, Seema Bawa, and Ravinder Kumar, "Evaluating Generative Adversarial Networks for Gurumukhi handwritten Character Recognition(CR)" has been published in **MISP 2022 at NIT Raipur**.

6. Kaur, Sukhandeep, Seema Bawa, and Ravinder Kumar "Script Identification in Handwritten Documents for Gurumukhi-Latin Script using Transfer Learning with Deep and Shallow Classifiers" **major revision** submitted in **Soft Computing (IF=4)**.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# List of Abbreviations

| | |
|---|---|
| AI | Artificial Intelligence |
| HTR | Handwritten Text Recognition |
| OCR | Optical Character Recognition |
| HOG | Histograms of Oriented Gradient |
| GLCM | Gray level Co Occurrence Matrix |
| SIFT | Scale Invariant Feature Transformation |
| SFS | Sequential Forward Selection |
| SBS | Sequential Backward Selection |
| SVM | Support Vector Machine |
| RF | Random Forest |
| KNN | K-Nearest Neighbor |
| HMM | Hidden Markov Model |
| ANN | Artificial Neural Network |
| DTW | Dynamic Time Warping |
| CR | Character Recognition |
| MLP | Multi layer Perceptron |
| FFNN | Feed Forward Neural Network |
| CNN | Convolutional Neural Network |

| | |
|---|---|
| RNN | Recurrent Neural Network |
| LSTM | Long Short Term Memory |
| GPU | Graphical Processing Unit |
| CPU | Central Processing Unit |
| GUI | Graphical User Interface |
| UML | Unified Modeling Diagram |
| LMCA | Lettres Mots et Chiffres Arabe |
| BRL | Black Run Length |
| NSCT | Non Sub Sample Contourlet Transform (NSCT) |
| PCA | Principal Component Analysis |
| QTLR | Quad tree longest run |
| DCT | Discrete Cosine Transformation |
| BLSTM | Bidirectional Long Short Term Memory |
| LBP | Local Binary Pattern |
| DHT | Distance-Hough Transform |
| MLG | Modified log-Gabor |
| EAST | Efficient and accurate scene text detector |
| CTC | Connectionist Temporal Classification |
| DCNN | Deep Convolutional Neural Network |
| RLSA | Run Length Smoothing Algorithm |
| DST | Discrete Sine Transform |
| HCCR | Handwritten Chinese Character Recognition |
| LM | Language Model |

| | |
|---|---|
| VP | Vertical Projection |
| DR | Detection rate |
| RA | Recognition Accuracy |
| SGD | Stochastic Gradient Descent |
| BHTRforAD | Bilingual Handwritten Text Recognition for Academic Domain |

# Chapter 1

# Introduction

This chapter introduces us to the different concepts of text recognition, evolution of text recognition, bilingual handwritten text recognition, machine learning and deep learning in text recognition, and various writing systems and scripts of the world. It also discusses the various phases of bilingual text recognition and the challenges faced by it. Further, numerous traditional and deep learning based approaches used for text recognition have been studied in detail.

## 1.1 Evolution of Text Recognition

Handwriting is the most popular way of communication in our day-to-day life. The usage of handwritten documents in public offices is the prime example of handwritten text [1]. People feel more comfortable in writing text as compared to typing. The advancement of technology has greatly influenced the ways of writing from handwriting typewriters, printing presses, etc. to word processor, e-mails, etc. The invention of pen computing devices and scanners in 1900 opened up the doors for digitization of text written on paper [2, 3]. The digitization of documents has gained tremendous popularity in the recent times. It helps in preserving the old manuscripts and documents safe for a longer period. Handwritten text recognition systems have been designed to make these digital copies more usable in terms of modifications and searching of content.

Text recognition has various application domains apart from digital simulation of handwrit-

ten text such as intelligent document recognition system, machine translation, image to text conversion, and image to speech conversion. The research in text recognition has evolved from machine printed text to handwritten text, video, and scene text recognition, etc. In the early days of 1940-50, template matching based text recognition systems were designed. The development of electronic tablets, led to the invention of commercial text recognition systems. Till 90s, text recognition systems were using statistical and structural information for recognition. After 1990, the research in text recognition has witnessed a major rise due to the invention of Artificial intelligence (AI) in pattern recognition. Many applications of text recognition like word recognition, signature verification, and writer identification have come into existence due to statistical and structural models. Language models and lexicons have further improved the accuracy of text recognition by considering semantic knowledge. Till the year 2010, the major focus of CR research community was on monolingual text recognition. However during the period 2005-10, this focus shifted from monolingual to multilingual text recognition with the incorporation of script identification module in text recognition. After 2010, a significant amount of work was done in the areas of cursive script recognition, scene text recognition, and multilingual text identification in video pictures. Further, after 2014, the deep neural networks gained popularity and provided a new direction to research in text recognition. These have helped to solve many issues of cursive, multilingual and complex scripts text recognition using a number of deep layers for feature extraction.

## 1.2 Text Recognition

Text recognition is the subarea of pattern recognition research field which recognizes the different forms of input like the text written in the form of images. It involves the study of attributes or characteristics of input data to determine the matching and mismatching in data. An efficient handwritten text recognition system is a computer based system which recognizes the text or symbols written by human hand or typed by machine. These systems are divided into two classes i.e., Online and Offline text recognition based on the method used for data collection. The offline text recognition system takes the scanned copy of text

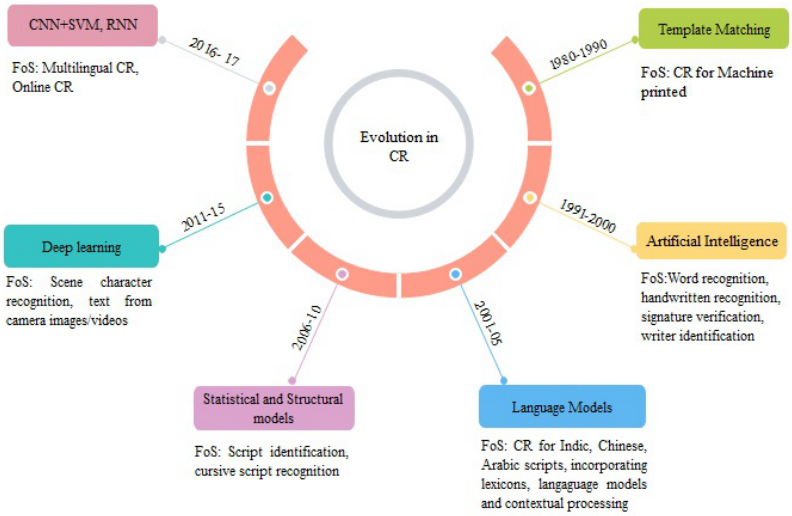



**Figure 1.1:** Evolution of Text Recognition

written in a piece of paper for recognition. On the other hand, online text recognition recognizes the text written through touch pad using stylus pen. It captures the temporal and static information in the form of 'X-Y' coordinates, stroke order, pen up and pen down, etc. This information makes the recognition process more easier than offline handwritten text recognition [4].

### 1.2.1 Handwritten and Printed Text Recognition

The offline text recognition is further categorized into printed text recognition and handwritten text recognition. Printed text recognition takes the machine typed text written using various font styles, while handwritten text considers the text written by human hands as input for recognition. Due to the wide range of individual handwriting styles, it is more difficult to recognize handwritten text. The printed text has stable quality of text due to single format used by machine from the start to end of the document, while handwritten text has variability in the text written due to the factors such as age, gender, mood and profession of the writer.

### 1.2.2 Monolingual and Multilingual Text Recognition

Further, text recognition systems can be classified based upon the scripts used by the text. Monolingual text recognition system recognizes the text written in one script or language,

while the multilingual system can recognize the text from more than one script and language [5]. The research in the field of monolingual recognition has attained a significant level for Indic as well as non Indic scripts. The presence of a number of writing styles and scripts in the world makes it difficult to use monolingual recognition systems [6]. Hence, the multilingual recognition system came into existence with its ability to recognize text from various scripts and languages. Traditional multilingual systems consider a number of text recognition systems for each script along with manual identification of scripts. It is a time consuming process [7]. However, the addition of automatic script identification sub-process in text recognition can make the recognition process faster and more accurate in multilingual systems.

### 1.2.3 Writing Styles and Scripts

A script is a graphic representation of a writing style using a particular collection of symbols [8]. A single script may contain a single language or a number of languages, each with a few subtle differences. There are mainly six writing styles in the world used by major scripts of the world. These are described as below:

1. Logographic System: A logographic system uses a set of symbols to represent a complete word. The scripts with logographic systems use thousands of character sets. Han, Kanji and Hanja are the major scripts included in this system. The character sets of these scripts included multiple short strokes giving them a complex look. The languages such as Chinese, Japanese, and Korean appeared under these. This writing style follows a particular direction of writing like either from left to write or from top to bottom.

2. Syllabic System: This system is used by Japanese where each symbol written represents a phonetic sound. These symbols are known as Kanas and are visually similar to Kanji script. It has two types of Kanas, i.e., Hirakana and Katakana which are less dense logographic characters.

3. Alphabetic System: This is the most commonly used writing style used in the world.

Many scripts like Latin, Greek, Cyrillic and Armenian are based on this system. It uses an alphabet which is a set of character to represent phonemes of a spoken language.

4. Abjads System: This system is quite almost similar to the Alphabetic system except each symbol represents the consonant sound. The other difference is the writing direction, i.e. from right to left. Further, it has characters with long strokes connected with each other giving an appearance of cursive script. The two major scripts that appear under this system are: Arabic and Hebrew.

5. Abugidas : Abugidas also use alphabetic based writing system including scripts from Brahmic family of scripts. These scripts have emerged from ancient Indian Brahmi script. The major scripts used in northern and southern parts of India are originated from Brahmi with slight modifications in each other. The most common feature of such scripts is the presence of header line. All the characters in a word are connected by this header line called 'shirorekha'.

6. Featural System: It includes features that make up phoneme generated by symbols and characters. Korean Hangual script is the prominent script based on this system. It is designed by mixing logographic Hanja and featural Hangual.

## 1.3 Bilingual Handwritten Text Recognition (HTR)

Bilingual HTR is a kind of multilingual text recognition system with only two scripts or languages. In a multilingual nation like India, people use two or more than two languages frequently for communication. The official documents of public domain includes text written in more than one language i.e., English and the regional language of a particular area or state [6]. Thus, it gave rise to the need of bilingual text recognition system. The coherent bilingual HTR system should recognize each piece of handwritten text in an unrestricted environment, regardless of the script, language, or writing style. It has seven sub-processes, viz. data acquisition, pre-processing, segmentation, feature extraction, script identification, character recognition, and post-processing. Figure 1.2 depicts the architecture of bilingual HTR. Data acquisition process captures the textual data in the form of images which is a further subject

for pre-processing. This precised data resulting from pre-processing phase is used to segment text into sub-parts like lines, words and characters. The inherent attributes extracted from these sub-parts are considered for script or language identification of the text. Based upon the script identified by script identification module, the corresponding OCR will be selected for character recognition. The post- processing sub-process generates the confidence measure of the output produced by the classifier. The different phases of bilingual HTR have been explained in the following subsections.

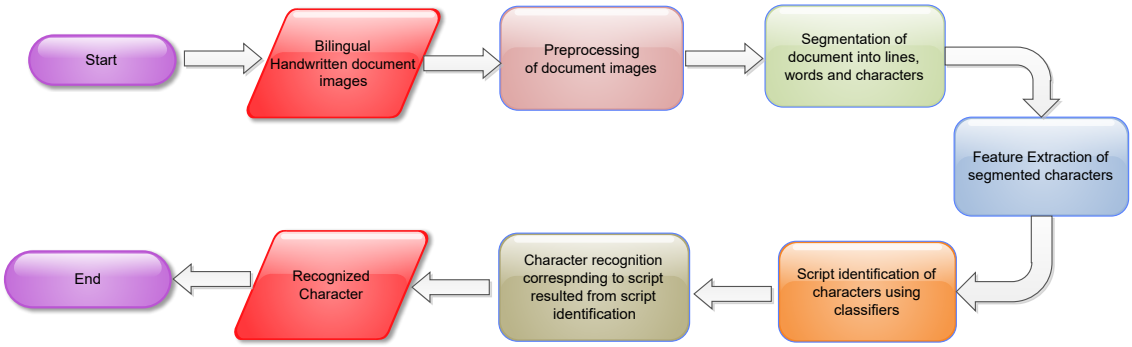



**Figure 1.2:** Schematic diagram of key modules of bilingual HTR

### 1.3.1 Data Acquisition

Data acquisition is the procedure of gathering textual information for recognition in the form of picture pixels or ”x-y” coordinates. Many digital devices like scanners, mobile phones, cameras, etc are used to capture the data. Large volumes of data are required to design the significant text recognition systems. It needs to cover the large variations in handwriting like writing styles based on age, mood and gender of the writer. The data collection process for online text recognition is somewhat different as compared to offline. It collects the data with temporal information at the various levels like stroke, character and word. For regional languages in Indic scripts, there is a lack of benchmark datasets for text recognition. The benchmark dataset provides a platform for comparison, evaluation and development of unbiased recognition systems for different writing styles [9].

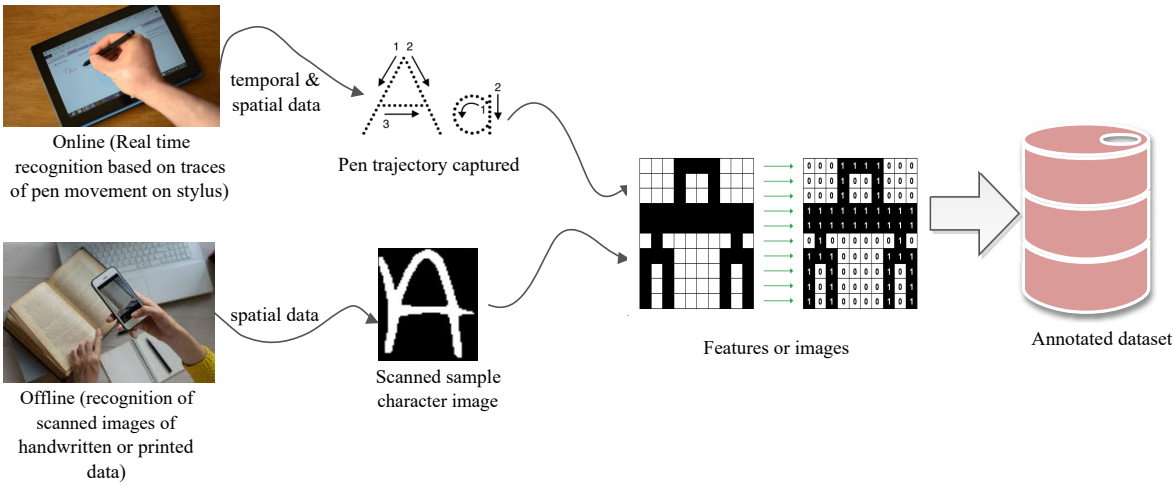



**Figure 1.3:** Data Acquisition Process

### 1.3.2 Pre-processing

The digital data collected during data acquisition process like images or x-y coordinates, usually, gets mixed with some unwanted data called noise. It can significantly reduce the performance of recognition system. The noise could be of two types; low level and high level. The low level noise appears due to software issues during scanning of the text while high level noise is considered anything present in the scanned document other than the required text. For example, a text recognition system has been trained for characters of a particular script. Now, if we input some different kinds of characters or special symbols, the output of recognizer will be unpredictable. This could be caused due to the presence of high level noise, i.e., unwanted characters. Thus, to avoid such unpredictable behavior of the recognizer, it is desirable to remove all such imperfections from the data.

Apart from noise removal, certain other approaches like slant and skew removal, thresholding, resizing, binarization, cropping, reference line detection, etc have also been performed. These are the essential steps to prepare data for the recognition process. Skew estimation is the most important sub-process. It detects the slope of text written with 'x-axis'. If the slope is not zero, it means the skew is present in the text. Similarly, slant is the angle between vertical axis and vertical strokes of the text. An ideal text has strokes written in a vertical position. The most challenging step in slant and skew correction is to determine the slant and skew angle [10, 11]. Most of the researchers have used projection profiles, Wigner Ville

distributions, cost functions and structural features using chain code to determine the slant and skew angle [12, 13, 14, 15].

The image acquisition phase captures the images in gray scale format using digital scanner. In gray scale format, data is represented using '0' and '1' while various shades of gray remain present in between 0 and 1. The process of storing these gray scale images into less storage intensity format, i.e., binary is called binarization or thresholding. Otsu is the most common approach used by the researchers in the past for binarization. Further, for cursive scripts, core region detection is also an important pre-processing step. The core region is the area between lower and upper baseline of the written text [16]. The correct core region detection could serve many purposes like skew and slant estimation, and determining the character height for contextual information [17]. The data collected during data acquisition may vary in size for each sample. Thus, during pre-processing, normalization of data is performed to make all samples of a uniform size.

### 1.3.3 Segmentation

The recognition of textual data is more efficient at the character level as compared to the line or word level. Hence, there is a need to isolate the characters from lines and words before transmitting to the recognition system for classification. Segmentation is the process of finding segmentation points in an image to decompose it into sub images for better recognition results. Equation 1.1 shows the segmentation process; where, image 'I' has been segmented into 'i' sub images using operator 'S' [18].

$$I(x,y) = S.i(x,y) \tag{1.1}$$

It is considered as one of the crucial parts of text recognition process. For text recognition, it is defined at four levels i.e. page layout analysis and segmentation, text line, word, and character. Segmentation is a more challenging task in handwritten text as compared to the printed form. In contrast to printed text, handwritten text has irregularly spaced lines and words, touching and curved letters, and variations in each writer's handwriting style, making segmentation a more difficult task.

Casey et al. have divided segmentation approaches into three categories i.e. classical,

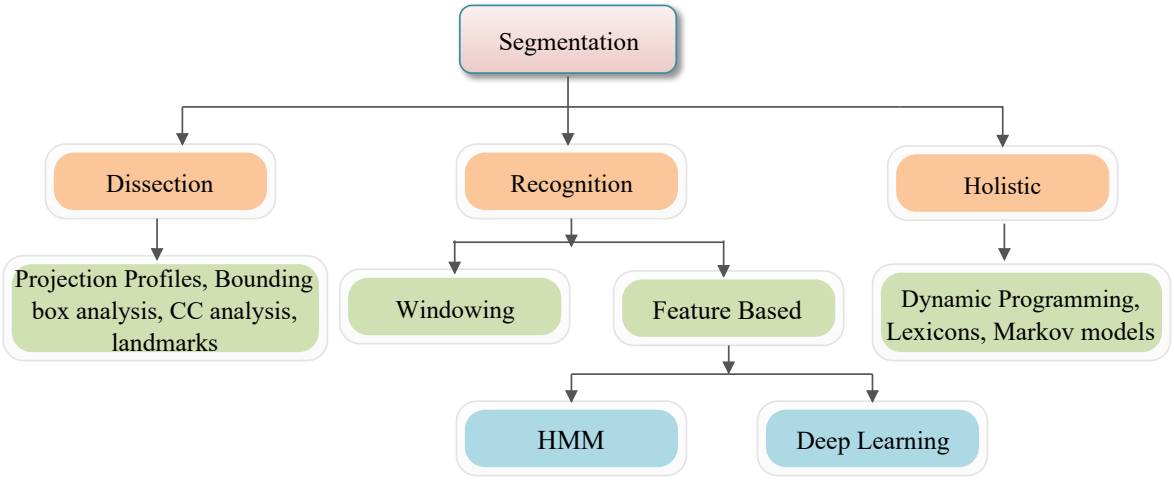



**Figure 1.4:** Taxonomy of Segmentation

recognition based, and holistic approaches [19]. The classical approaches use certain attributes of written text to segment the text into meaningful components. Projection profile based approaches are the most common examples of classic approaches for segmentation. In recognition based approach, a recognition module runs simultaneously with segmentation phase to find the optimal cut points. Some common techniques used in these are Hidden Markov models, dynamic programming and neural networks, etc. Lastly, holistic based approaches are designed by combining dissection and recognition based techniques. These approaches consider recognition module to select the optimal segmentation point resulting from dissection method.

### 1.3.4 Feature Extraction

In any pattern recognition problem, feature extraction has the most significant role for classification of data. Features are the variables representing the characteristics or attributes of any specific data. It is crucial job to design an efficient feature set for the classification task. The feature extraction process consists of four main sub processes, i.e., feature generation, feature selection, feature reduction, and feature evaluation. Feature generation process designs the features using statistical and structural properties of data. In text recognition, features extracted are categorized into two categories based on the level of information extracted, i.e.,

low level features and high level features. The low level features consider the neighboring points in the data to extract information like area, 'x-y' coordinates, tangents, etc. These lower level features are combined at a large scale to calculate the high level features of data like crossings, dots, and loops [20]. Various methods such as Gray Level Co Occurrence Matrix (GLCM), Histograms of Oriented Gradient (HOG), Scale Invariant Feature Transformation (SIFT) are available to extract the low level features.

These extracted features, based upon the feature extraction process, are further divided into traditional features and deep learning features. Traditional features consider human expertise and traditional approaches like Gabor filter, HOG, SIFT, etc. to represent the geometrical and surface properties of the text. It requires to identify the appropriate region for feature extraction and then the corresponding feature extraction approach. Hence, the results of classification using such features are highly dependent upon the type of pattern used and approach applied for feature extraction. Statistical and structural features are the two major categories of such features. The statistical features use zoning, projection profile based approaches, crossing and distance based feature extraction approaches. These features are noise and deformation tolerance. Structural features use geometrical properties of text to describe the shape of written text. The features extracted using curves, edges, loops, branches, aspect ratio appear under these. Another category found on the basis of transformations is texture features. It includes Gabor filters, GLCM, Gradient, cosine based and wavelet transformation based features.

The latest trend in feature extraction is deep neural network based features. These networks are capable of automatically extracting the high level features of pattern using a number of deep layers. Convolutional Neural Networks are highly employed as feature extractor for pattern recognition problems.

The extracted features may contain some irrelevant and redundant data. Thus, a filtration process is required to remove such irrelevant data from the designed feature set during the feature generation process. Some of the researchers have found Sequential Forward Selection (SFS) and Sequential Backward Selection (SBS) approaches significant for feature selection [21]. Further, filter and wrapper approaches have been designed using some kind of ranking algorithms [22]. Sometimes, the extracted feature set has large dimensions which

makes the task of learning more complex. Hence, to reduce the dimensions of feature set without losing the relevant information, feature reduction process is considered essential. The approaches based on principal component analysis are widely adopted for dimensionality reduction [23].

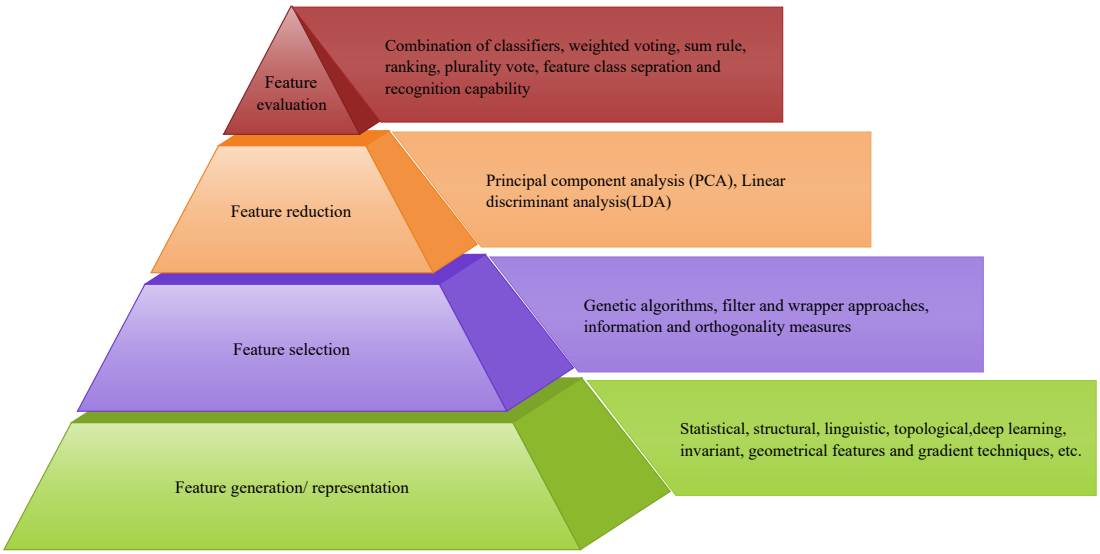



**Figure 1.5:** Various steps for Feature extraction process

The accuracy of recognition process highly depends upon the features used. In text recognition, it is difficult to choose the promising features due to large variations in individual handwriting style. Hence, it gives rise to the need of some kind of feature evaluation criteria for extracted features [24]. The most commonly used approaches used are weighted voting, classifier combination, plurality vote, and ranking for feature evaluation [25, 26].

### 1.3.5 Script Identification

Script identification is the process of determining the script of text written using some features which distinguish it from other scripts. The distinct spatial distribution and visual characteristics of script are represented by these features. Hence, the motive of script identification is to discover the features of text and then classifying the text accordingly. Script identification is employed at the various levels like page, paragraph, text line, word, and

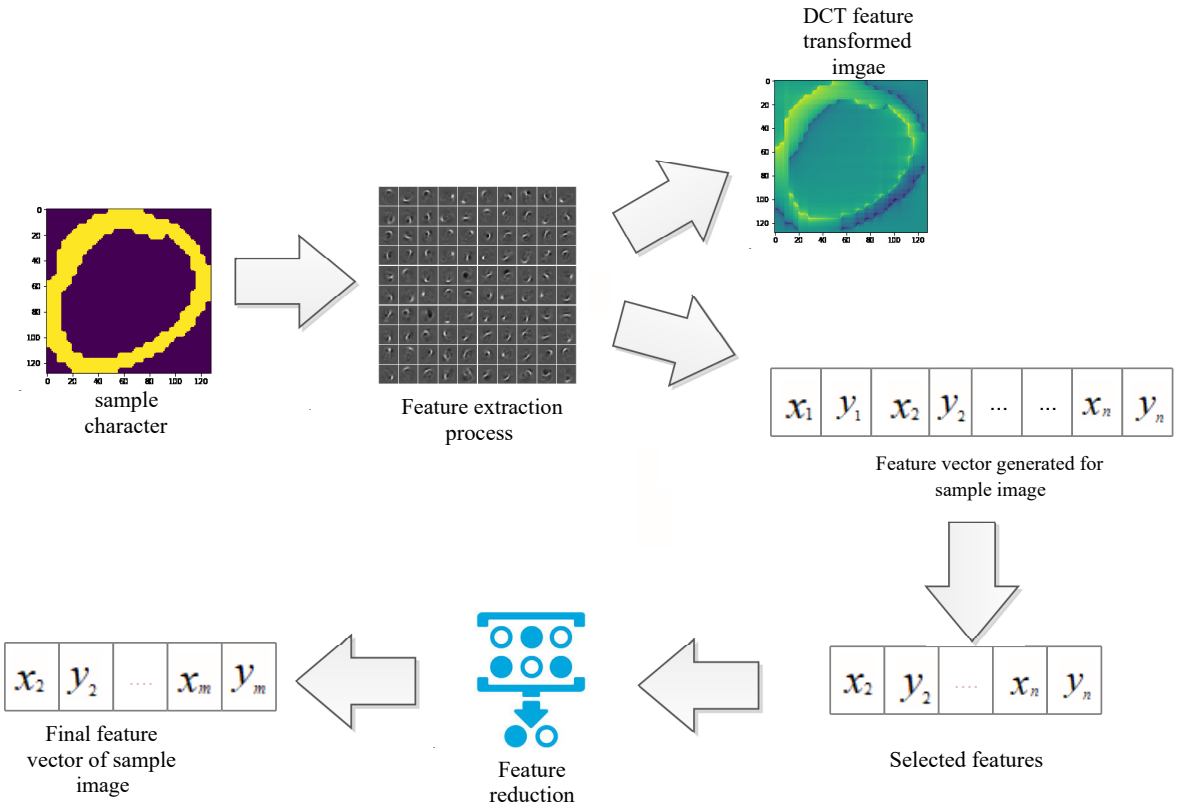



**Figure 1.6:** Feature extraction process of image data

character. Script identification approaches are categorized into two classes based upon the features and approaches used, i.e., structure based and visual appearance based.

Structure based methods use stroke structure, connected component, and writing style of scripts to identify each script. Connected component analysis has been used to analyze the morphological attributes of script. Further, there are some scripts having characters connected by line in a word. In such a scenario, connected component analysis based methods are applied for script identification. The textual symbols of each script have been considered as the unique features for identification at the page level. It includes discrete characters, adjoined characters, etc. These methods require larger blocks of text to extract the information required for script identification. However, in the multilingual text, a smaller block of text has been taken for script identification. In multilingual country like India, text document contains different scripts at the line, word and character levels. To identify the script of a paragraph or text line, horizontal projections, neural networks and texture features have been used. Script identification at the word and character levels is more difficult than the line

level. Because the regions available for feature extraction during the word or character level are not sufficient to identify the script.

The appearance based methods consider that each script has a different visual appearance like shape of character set, word and sentence formation etc. Hence, a glance at the written text provided that a casual observer can differentiate without performing any component analysis operations. The most common approaches used under these are the vertical and horizontal projection profiles. It further considers script identification as a texture analysis problem.

### 1.3.6 Character Recognition

The recognition process identifies the unknown pattern and assigns a class label to that pattern. Suppose, there is an 'n' dimensional feature vector extracted from a pattern, i.e., $V_i = v_1, v_2, ..., v_n$, the recognition model will design a classification function 'f' which maps the feature vector to corresponding class of pattern, i.e., $c_i$ from set of classes, i.e., $C$.

$$f : (v_i \rightarrow c_i | c_i \in C) \tag{1.2}$$

The recognition models are divided into two categories, i.e., shallow architecture based models, and deep architecture based models [27].

Shallow architectures are the traditional machine learning based classifiers like Support Vector Machines (SVMs), Random Forest (RF), K-Nearest Neighbor (KNN), Hidden Markov Model(HMM) etc. These classifiers consider features extracted from traditional feature extraction approaches to train the recognition model. The following are the examples of shallow classifiers:

1. Hidden Markov Models (HMM): HMM is commonly used in online text recognition tasks. It computes the probability of a particular observation sequence for a model having various state transitions probabilities and observation symbol probabilities. Every class of data is modeled with an independent HMM. The output class for an unknown pattern will be the class having HMM with the highest probability [28]. These recog-

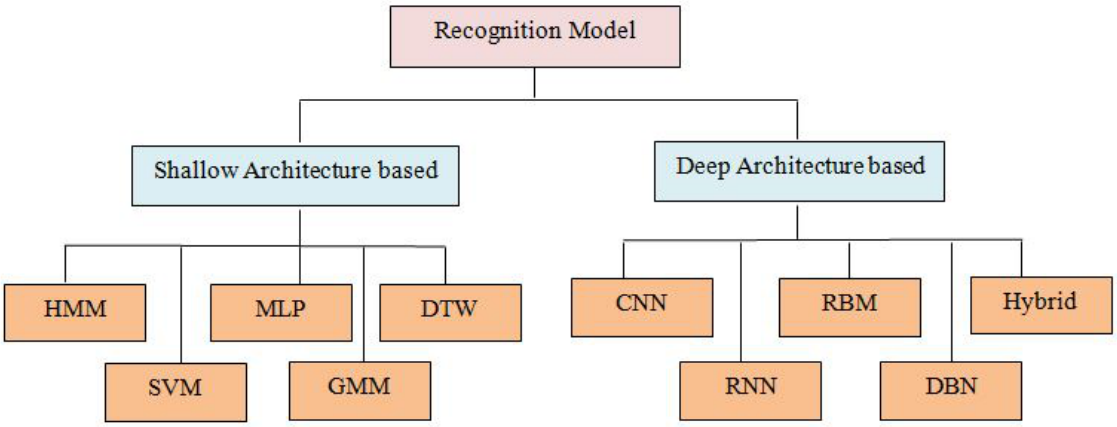



**Figure 1.7:** Hierarchy of various recognition models

nition models can easily tolerate the insertion, deletion and substitution errors in data. In text recognition, two types of HMM models are designed : model discriminate and path discriminate. In model discriminate, for each word in the lexicon, a specific model is trained. It requires sufficient samples for each word to design a training model accordingly. The other approach, i.e., path discrimination uses a single HMM for all the words. It models each character using smaller group of states.

2. Support Vector Machines (SVMs): These provide the supervised learning of data for classifications and create decision boundaries called hyper planes for classification of data into different groups with the help of training samples. These machines use linear and non linear functions to design the hyperplane corresponding to the type of pattern. The mathematical representation of linear functions is as below:

$$y(x) = S.x - a \tag{1.3}$$

where, 'y' is the vector normal to hyperplane; and 'a' is the displacement relative to the origin [29]. For data with high dimensional space, linear function will not be able to generate efficient decision boundaries. In such cases, a new hyperplane and support

vectors are needed to map the data to higher dimensional space, i.e.

$$y(x) = \sum_{i=1}^{n} \gamma(i)c(i)K(x_i,x) - b \tag{1.4}$$

It uses non linear kernel function, i.e., $K(x_i,x)$.

3. KNN: These are the supervised learning methods where the distance between training samples is used for classification. KNNs find the closest group of training objects based on the distance between these objects and assign the most common class among their 'k' distance nearest neighbors. Suppose, there is a 'x' instance from test set, KNNs will find the k-nearest neighbors of this object. The class which is most common to these nearest objects will be assigned to the test instance. The commonly used distance measures in conventional KNN are Euclidean distance, cosine similarity, Minkowsky, correlation, and chi square. Many researchers have designed new KNNs with modifications into the distance measure function.

4. *Artificial Neural Networks (ANNs)*: These are the information processing systems where number of simple cells or neurons are interconnected through weighted connections. The connections of these neurons are inspired from biological neurons as each neuron derives its input and output from other neurons. During the learning of networks, the weight of each neuron get updated until the best weights are produced. After training, these stable weights have been used for classification of data. Backpropagation mechanism has been used to update the weights. In backpropagation, initial weights are propagated to the network in forward direction from input to output. After that, the error between the observed output and actual output is propagated to update the weights and biases. The mathematical formulation of ANN is as follows:

$$y = f(w_1 i_1, w_2 i_2, ..., w_n i_n) \tag{1.5}$$

where, $i_1, i_2, ..., i_n$ are the inputs at each node; and $w_1, w_2, ..., w_n$ are the updated weights at each node [27].

5. *Template matching*: Template matching methods use the skeleton or template of a pat-

tern for a reference to recognize the input pattern without using any feature extraction approaches. This approach is considered the simplest due to its independence to training data and easiness in handling the manual conflicts. It uses segmented sub parts of character pattern to map the template pattern. Hence, it requires to identify the segmentation points before recognition using global normalization and re parametrization [30]. Dynamic Time Warping (DTW) is the most popular approach used for template matching in CR.

The sallow architectures are unable to represent the real world problems like human visual system due to their limited representational power. Such problems have complex layered information which needs powerful models with many deep layers to extract it. Deep architectures used in deep neural networks can extract such rich sensory inputs with the help of hidden layers. These models are capable of automatically extracting the features using unsupervised learning.

The idea of deep network comes by adding the number of hidden layers in the traditional artificial neural networks, i.e., Multi Layer Perceptron (MLP), and Feed Forward Neural Network (FFNN), etc. However, initially, these models faced the issues of learning weights, local optima in backpropagation, and processing power to train the model. There was a rise in the use of deep networks after Hinton found a solution to the problem of optimization in '2006' [31]. He has designed Deep boltzman Machine using a number of FFNNs. After this, many other researchers used these machines for speech recognition and other pattern recognition problems [32]. CNN, RNN, and LSTM are the most popular deep networks used for text recognition. These have been explained as follows:

1. *Convolutional neural networks (CNN)*: Theses are the most popular models among the deep networks. The structure of these models is like FFNNs except the three different layers, i.e., convolutional, pooling, and fully connected layers [33]. The convolutional and maxpooling layers are connected alternatively and followed by the fully connected layer. These layers extract maximum information of input pattern in the form of certain features. The first layer extracts intersections, while the second layer extracts the edges and so on. As this information passes to higher layers, it becomes more invari-

ant. The last fully connected layer of the network represents it in the form of vector. This vector is passed to an MLP classifier for generating the output.

*Convolutional Layers:* These use convolutional operations on the small portion of image with small learnable weights to extract the features of pattern for that plane. These weights are shared by all the feature maps of that plane. For this, a filter called kernel of size M×M is used on the input plane. This filter performs convolution operation by sliding over the input image. Horizontal and vertical distance used to move the window on input plane is called stride. The mathematical formulation of the output feature map at convolutional layer l is calculated as:

$$o^l_{i,j} = f^l(\sum_{m=1}^{k}\sum_{n=1}^{k} w^l_{m,n}.o^{l-1}_{i+m,j+n} + b^l) \tag{1.6}$$

Here $w^l_{m,n}$ is the convolutional kernel at layer l; and $o^{l-1}_{i+m,j+n}$ represents the feature map of previous layer; and $b^l$ is the bias input. The function $f^l$ represents activation function responsible for generating the non linearity in the network.

*Pooling Layer:* This layer is used to reduce the complexity of network by generalizing the feature map generated by convolutional layer. It reduces the dimensions of the feature map based on the size of mask used. It does not alter the feature map, instead it uses some kind of pooling. Like the max function used in max pooling, it extracts the maximum value from feature maps to get the higher level features which are robust and translation invariant. This process reduces the learning time of the network. The mathematical formulation of max pooling is:

$$F^p_{i,j} = max(f^{p-1}_{1+k(i-1)+k(j-1))}, .., f^{p-1}_{k_i+k(j-1))}, ...) \tag{1.7}$$

*Classification:* The last layers of CNN are fully connected layers which consist of high level features. These layers use the one dimensional flatten feature vector as input and generate the output class. It uses softmax classifier which is a simple logistic regression for generating the probabilities of each output class [34, 35, 36, 37, 37].

2. *Recurrent Neural Networks (RNN):* These are used to learn the sequential data. These

implement the sequential behavior with the help of loop present on hidden layer computational units. For the implementation of RNNs, correlation is needed among data points. The labeling of input sequence in RNN is done using a set of rules corresponding to the past sequence. For this, it uses feedback connections to retain the past sequence calculations as input for next iteration. The hidden state $S_n$ is represented as:

$$S_n = UI_n + WS_{n-1} + b \tag{1.8}$$

Here, $S_{n-1}$ represents the output of previous hidden unit while $I_n$ represents the current input [33, 38]. The RNN output i.e. $O_n$ is produced using linear transformation of hidden layer output.

$$O_n = VS_n + c; where \tag{1.9}$$

3. *Long Short Term Memory:* RNNs suffer from the problem of vanishing gradient. Sometimes, for the complex and large inputs, it is difficult for RNN's hidden units to retain the complex information for a long time. Hence, the information required for a longer time starts vanishing from the cells memory after some time. This issue has been resolved by designing the LSTMs through the replacement of RNN encoders with an internal memory structure. LSTM uses three gates i.e. input, output and forget gate, it helps in modeling the large temporal dependencies. The input gate determines which input information should be stored in the current hidden state. The output gate determines which information from the hidden state should be included in the output at the current time step. The forget gate decides which information from the previous hidden state should be "forgotten" or excluded from the current hidden state. The following composite functions calculate the hidden layer function as:

$$i_t = \sigma(w_{xi}x_t + w_{hi}h_{t-1} + w_{ci}c_{t-1} + B_i) \tag{1.10}$$

$$f_t = \sigma(w_{xf}x_t + w_{hf}h_{t-1} + w_{cf}c_{t-1} + B_f) \tag{1.11}$$

$$c_t = f_t c_{t-1} + i_t \tanh(w_{xc}x_t + w_{hc}ht - 1 + B_c) \tag{1.12}$$

$$O_t = \sigma(w_{xo}x_t + w_{ho}H_{t-1} + w_{co}c_t + B_o) \tag{1.13}$$

$$h_t = O_t \tanh(ct) \tag{1.14}$$

where i, o and f correspondingly denote the input, output and forget gates. $\sigma$ refers to the logistic sigmoid function and c stands for cell. [39]

### 1.3.7 Post-processing

The post processing phase is used to evaluate the character recognition results generated from the recognition model. In this, the classified text from recognition models is matched with some predefined sets of labels for efficient results. It helps in reducing the errors in classification. Kukich et al., have found three types of errors in text recognition problem: non word, isolated word, and context dependent errors [40]. To improve the classification results, some kind of confidence measure is required to access the recognition results. Many strategies prevail to find the confidence measure such as posterior probabilities, maximum likelihood ratios, etc. The mathematical representation of confidence measure is as below [41]:

$$p(t|h) = P(h|t)P(t) \tag{1.15}$$

The four major factors which affect the post processing results are: confidence measure, lexicon size, statistical language model, and searching strategy [42]. RNNs are widely used as language models for post processing of speech and text recognition applications [43, 44].

## 1.4 Domain based Bilingual HTR

Domain based handwritten text recognition system recognizes the text from a particular domain like academic, health, transport etc. However this work considers only the documents of academic domain for text recognition. Handwritten documents are used by a large number of public domain offices for the maintenance of records. Each domain has a different text to be used, writing style of users, qualification of users, different page style and structure, and different categories of users. Thus, it is quite challenging to develop a single text recognizer

for all the domains. Keeping this problem in view this work considers only the text written for academic domain. Of all these domains, academic domain is the one which highly uses bilingual text in official documents. In the Punjab state, Punjabi and English languages representing the Gurumukhi and Latin scripts respectively are used in official documents. As an example, the document image under this domain is shown in Figure 1.8.

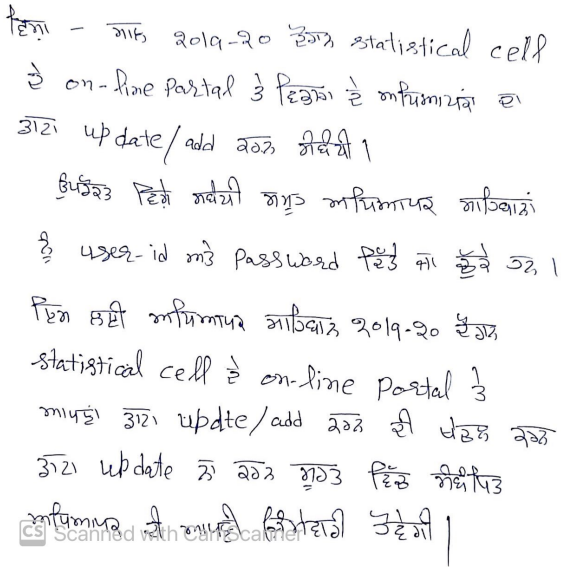



**Figure 1.8:** Sample of Bilingual document from Academic domain

Bilingual text recognition systems are required to convert such documents into machine editable form. This recognition system is designed by using all the sub processes of a bilingual text recognition corresponding to academic domain data. It can also be used for other domains by training the system corresponding to the required domain dataset.

### 1.4.1 Gurumukhi Script

Gurumukhi script was standardized by the second Sikh Guru, i.e., Shri Guru Angad Dev ji in the sixteenth century. The word 'Gurumukhi means' "from the mouth of Guru". Gurumukhi script from the Indic scripts category is the base for Punjabi language [45]. The Punjabi language is spoken across the countries like India, Canada, America, Australia, New Zealand, India etc. Gurumukhi script has some resemblance with the Devanagari and Bangla scripts. The writing direction of script is from left to right and in top to down approach. The character set of Punjabi language includes 35 basic characters, 12 vowels, 6 additional modified

consonants, and 3 half characters. The character written in Gurumukhi script has a header line present at the upper part. Gurumukhi script is a cursive script where characters in the word are connected by header line. Gurumukhi word is formed with the combination of character and vowel. The word written in Gurumukhi script is divided into three horizontal zones, i.e., upper zone, middle zone, and lower zone. The region above header line is called upper zone, where most of the vowels appear. The area between header line is the busiest zone called middle zone, where consonants and some parts of the vowels remain present. The lower zone is the area below middle zone containing vowels.

### 1.4.2 Latin Script

Latin script belongs to the alphabetic writing system of the world. It is also called the Roman script. The most popular languages appearing under this are English, Italian, French, German, and Spanish, etc. English is considered as the standard language across the world for communication [8]. It has the simplest character structure composed of few lines and arches. It has 26 capital letters and an equal number of small letters in the character set. Due to the effect of colonial rule by the British, English has been accepted as a supporting language along with the regional languages across the world. English language is a cursive language as characters in a word are connected at the lower line.

### 1.4.3 Numerals and Special Characters

There are mainly 10 symbols used in the Arabic numeral system to represent the numbers. These symbols can be used individually or in combination to represent the number. These are called Arabic numerals. These were introduced in the tenth century in Europe and America by the Arabic speakers. Around the 13th century, the European mathematical circle had accepted the western Arabic numerals; and by the 15th century, its usage became common. Apart from alphabets and numerals, there is another category of characters called special characters. Punctuation marks and some symbols appear under this category. However, for the purpose of this research work, the special characters available on typical US keyboard and widely used in official documents have been considered.

## 1.5 Deep Learning and Machine Learning in Text Recognition

Deep learning is the class of machine learning which uses many stages of non linear information processing for pattern classification and feature extraction. Before deep learning, the traditional machine learning architectures used a single layer of non linear transformation for converting the raw data to problem specific feature space. The simple architecture of machine learning models is able to solve many simple and well constrained problems. But, it lacks in modeling and representing the complex real world problems like human speech recognition, scene text recognition, etc. Deep learning has been inspired by human visual system which is hierarchical in nature. Machine learning models need explicit methods for feature extraction and classification. However, in deep learning, the pre training steps extract the structure and regularities in data in the form of features using a large number of unlabeled training data. Further, to fine tune the data, supervised learning is performed using the labeled data.

### 1.5.1 Transfer Learning

Deep learning models have the only disadvantage of high training cost of network. These networks require huge training data which increases the number of trainable parameters of the network. Some researchers have developed parallel computing techniques to decrease the training time. These include GPU based systems which speed up the training of network. The recent trend emerging in deep learning is transfer learning which optimizes the training cost of network [46]. The example given below clearly defines transfer learning:
Suppose, a large dataset $D_1$ is used to train a model M using deep neural networks for a source task. There is another dataset $D_2$ which is smaller than $D_1$. There is another model N which is to be trained using dataset $D_2$. But the dataset $D_2$ is unable to train the model N from scratch due to its small size. Transfer learning provides the way to reuse a part of pre-trained model M to train the model N for dataset $D_2$. However, both the datasets should belong to the same domain.

#### 1.5.1.1 Need of Transfer Learning

The classification models for the real world problems like text and speech recognition demand larger deep networks with a number of deep layers. The training of such deeper networks demands a huge amount of training data for optimization of hyper parameters. The optimal way is to use the weights of pre-trained models on the large dataset, for research problem having smaller dataset with the help of transfer learning. The other major reason is the training time needed to train deep network based models from the scratch. These models need large processing power to train a large number of trainable parameters. The initial layers of such networks extract the low level features of the pattern like edges, etc. The weights of these layers can be reused from one dataset to another of the same domain. Hence, to decrease the training cost of the network, weights of lower level layers are reused in transfer learning; and only higher level layer weights are optimized corresponding to the dataset.

#### 1.5.1.2 Techniques of Transfer Learning

There are numerous techniques to implement transfer learning using pre-trained models. Each pre-trained model has a different architecture. The selection of such models depends on many factors like how similar is the source and target domain data, size of the dataset, computational resources, etc. These models are trained on Imagenet dataset having 15 million images with 22000 class labels. Thus, such a large dataset is sufficient to train an accurate classifier. These models have been investigated as hereunder:

1. *AlexNet:* AlexNet was designed in 2012 by Hinton. It is comprised of five convolutional layers, followed by maxpooling, dropout, and batch normalization layers. It has three fully connected layers. The inputs to this architecture are the images of size $227\times\ 227\times 3$. To perform transfer learning, the last two fully connected layers can be replaced. In this network, certain improvements were made through are the use of augmented dataset and training on multiple GPUs. Further Relu was used in place of Tanh which helped to solve the issue of vanishing gradient. Furthermore, the use of dropout layer and data augmentation helped in solving the problem of over fitting of network.

2. *VGG16 and VGG19:* VGG16 has more deeper layers than AlexNet. It consists of 13 convolutional layers along with maxpooling layers. Due to its denser architecture, it requires three times more trainable parameters and computation power than AlexNet. VGG is designed for input images of size 224×224×3. Like AlexNet, the last 16th convolutional layer is replaced with the adaptation layer having two convolutional layers for transfer learning. Thus, the trainable parameters from convolutional layers 1 to 15 are kept frozen; and only the parameters of last fully connected layers FC16 and FC17 are trained. VGG19 processing is similar to that of VGG16 except that it has 16 convolutional layers and 3 fully connected layers resulting in 19 layers in all.

3. *ResNet50 and ResNet150:* These are the CNN models designed to resolve the vanishing gradient and degradation problems. It uses the concept of residual learning in layers. It is identity mapping which does not have any parameters. It simply uses skip connections to add the output of previous layer to the output of layers ahead. It helps in training the network with a large number of deep networks. Hence, the authors of ResNet had tested it with hundreds of layers. ResNet with 152 layers had less number of trainable parameters than VGG due to residual mapping.

4. *LeNet5:* LeNet is one of the oldest pre trained model proposed by Yann LeCun in 1998. They have used it for image classification of handwritten and printed characters. It is a five trainable layer model consisting of 3 sets of convolutional and average pooling layers. It takes an input of size 32×32 grayscale images [47].

5. *GoogleNet (or Inception v1):* It was designed in 2014 by Szegedy and had won the ImageNet challenge. It uses the inception module. Each inception module has two layers and six convolutional blocks. It acts like a multi level feature extractor and uses less number of parameters than other networks. It has twelve times less parameters than AlexNet.

### 1.5.2 Ensemble Machine Learning

Ensemble learning is the way to revamp the recognition accuracy of classifier by combining different classification algorithms together. It can be performed in various ways such as

exploiting the training parameters of classifiers, considering various subsets of classifiers, manipulating the input features and output of classifiers. There are two major conditions to design an ideal ensemble classifier. The base classifiers need to be highly diverse and the prediction performance of ensemble classifier needs to be higher than the base classifiers. To combine the complementary nature of different classifiers, ensemble based classification is the optimal solution.

#### 1.5.2.1 Need of Ensemble Machine Learning

The three major reasons led the research community of machine learning to design the ensemble of machine learning approaches, i.e., statistical, representational, and computational. Sometimes, the statistics of training data are not correspondingly sufficient to the hypothesis space. Hence, the results generated by each learning algorithm remain the same. To prevent this wrong classification, ensemble learning uses the average results of each individual base classifier. Some algorithms, such as decision trees and neural networks, are challenging to train for large numbers of training samples because of the local optimum problem. It further increases the computation cost of algorithm. Ensemble learning provides solution to this by running many local searches with many different starting points. In certain cases, simple machine learning models are not able to represent the true function for training data. An ensemble classifier uses weighted sums of hypotheses to expand the space of representation function.

#### 1.5.2.2 Techniques of Ensemble Machine Learning

Three approaches are most commonly followed to design the ensemble classifier using different machine learning algorithms. These are explained as hereunder:

1. Voting: In, Voting based ensemble, each classifier's predictions are treated as votes. The final prediction of output class will be the one having maximum votes. Two types of voting based ensembles are used, i.e., majority voting, and weighted voting. The majority voting outputs the prediction class having more than half of votes from individual classifiers. The weighted voting assigns a weight to the better model and runs it multiple times.

2. Bagging: It is also called bootstrap aggregation. The bagging-based strategy creates homogeneous learning models that correspond to the sub samples that are designed at random from the training data. Suppose, we have a training dataset of size n; and N training subsets have been chosen randomly from this data. Now N individual classifiers will be generated corresponding to each subset. Each subset has an average of 63.2% of the original data and certain repetitions. This ensemble approach is considered to be the best for unstable base classifiers like neural network, decision tree, etc. where, a little modification in training data can highly affect the learning algorithm.

3. Boosting: This ensemble approach also uses the subset of training data and applies homogeneous models on these subsets. However, the major difference lies in the process of choosing the training subsets. It considers the classification results generated by learning algorithm in each iteration and assigns weights to samples using some kind of cost function. The wrongly classified training samples are assigned with higher weights to boost their performance.

## 1.6 Research Gaps

The various research gaps found after reviewing the process of text recognition approaches are explained as fellows:

1. *Exploring CR in hybrid documents, videos and camera images:* The hybrid documents include handwritten and printed text, as well as graphic or pictorial information. It is a quite challenging task to process data with large variations. Further, more effective preprocessing and segmentation algorithms are needed for text detection and localization in video and camera-captured pictures. Many researchers have explored deep architectures for scene text recognition and found promising results for non-Indic scripts. An equilibrium between model performance and training costs must be found. Further, for Indic scripts like Telgu, Punjabi and Tamil, text recognition in hybrid documents is required to be explored.

2. *Feature extraction using deep architectures:* The traditional feature extraction ap-

proaches demand human expertise to design an approach for feature extraction. Additionally, these approaches are unable to extract the high level features of pattern needed to classify the complex patterns. In text recognition, there are characters of the same shape in certain scripts. These features fail to identify such complex patterns. However, deep network based features automatically extract the features using deep layers of abstraction. For multilingual text script, independent features can be extracted through deep networks. Further, the traditional classifiers when used with these features can considerably raise the recognition accuracy of the model.

3. *Segmentation:* Segmentation in handwritten documents, is more challenging as compared to the printed text. The irregular page size, touching, skewed, curved and closed text lines present in handwritten documents make this task more complex. The actual shape of character gets distorted in segmentation of touching and overlapping characters. Thus, some kind of linguistic features are needed to remove the errors of segmentation.

4. *Improving various performance measures:* Accuracy is the most commonly used performance measure in text recognition. However, for multilingual text recognition, some other parameters like size of documents, font style, mode of text, etc. are also significant. The CPU time is also one of the prominent parameters to evaluate the various recognition algorithms.

5. *Classification using deep and shallow architectures:* The complimentary nature of different classifiers can be combined through a certain strategy to improve the performance of recognition process. The training samples wrongly classified by one classifier can be accurately recognized by another classifier. Ensemble classification is the prime example of such a classifier combination approach. The two most common strategies combining different architectures are cascaded and parallel.

6. *Text recognition issues in Indic scripts:* In majority of the Indian scripts, a header line connects the characters in a words. It makes the task of character segmentation difficult. The writing zone is further split into three subzones, namely the upper zone,

middle zone, and bottom zone. The zone identification for segmentation of data is a crucial task. The other major issues include matras present in the upper and bottom zones of characters, vowels connected with characters and confusing structure of vowels.

7. *Script specific observations:* Unlike the Indian scripts, non Indian scripts such as English and Arabic have achieved efficient accuracy in the matter of text recognition. In India, there are 22 official regional languages which need to be explored for text recognition.

8. *Issue of similar and confusing characters:* Certain scripts have characters of similar shapes. The identification of such characters in multilingual CR is really a tedious task. Structural attributes fail to distinguish such character pairs. Further, during the early stages of recognition like segmentation, the candidate attributes of such characters get distorted. Therefore, some kind of early pre-processing stage is needed to maintain the actual shape of such character pairs.

9. *Benchmark datasets:* Lack of sufficient study for regional languages in multilingual CR has been caused by the absence of any benchmark dataset. For multilingual recognition, we need a corpus having character sets from different scripts with large variations in writing styles. The dataset should consist of optimal number of samples capable of maximizing between class variations and within class variations.

10. *Incorporating semantic and linguistic corporation:* In multilingual CR, the popularity of artificial intelligence based approaches like knowledge based systems have raised the need of semantic and linguistic information. It helps in evaluating the recognition results and corrects the wrongly classified patterns. However, it is of great significance to choose the best searching strategy, language model and confidence measure.

11. *Deep architecture for multilingual CR:* In multilingual CR, a good number of classes exist in the training data which need deeper architecture to classify such a complex data. However, computational complexity is the main hurdle in training deep architectures. Some pre-trained networks can be used to improve the training time and cost of

deep networks.

## 1.7 Problem Statements

Many real-world computer vision applications, including content-based image retrieval, digital libraries, document image processing, multilingual text recognition, scene, video, and natural image text recognition, rely on CR. There has been tremendous development in the field of technology which has highly influenced the research trends in CR for regional languages. There are numerous scripts and writing systems in use today. Generally, the people use two or more writing styles for day-to-day work. In a multilingual country like India, the public office documents are available in two languages, i.e., English and the regional language. Thus a dire need is felt to develop a multilingual text recognition system.

To convert the handwritten multilingual manuscripts into machine editable form, the most important phase is the text segmentation at the line, word and character level. The presence of irregular spacing, curved, skewed and touching text, and large variation in individual writing styles makes the segmentation of handwritten text more difficult as compared to the printed text. Further, results of segmentation process vary with the use of different scripts, digitization method, writing style and mode, image type and size.

In multilingual text, certain similarities exist in various scripts within classes and between classes. A single feature set and classifier is not sufficient to classify such data. As each feature set and classifier provides different results for different patterns, thus, there is a need to combine the complementary nature of different classifiers and feature sets. Further, to process multilingual text, a sub process called script identification is required. Script identification can be performed at the page, paragraph, line, word and character level. In public office documents, some words contain characters from different scripts. Hence, it gives rise to the need for script identification at the character level. Moreover, the feature extraction is more efficient and less time consuming at the character level as compared to the word and line level. After script identification, the corresponding OCR is required for text recognition. Due to an insufficient amount of regional data, it is challenging to employ deep learning techniques for text recognition for regional languages. Thus, some sort of transfer learning

and parallel processing approaches are needed to utilize the power of deep networks for text recognition of regional languages. The potential methodologies used under transfer learning are fine tuning, domain adaptation and feature extraction while for parallel processing are data parallelism, model parallelism and pipeline parallelism. In fine tuning, pre trained models are fine tuned to new tasks while for feature extraction we use the activations of pre-trained models as features for a new model. In this we freeze the pre-trained layers and extract features from intermediate layers, which are then used as inputs for a new model. Further for domain adaptation, a model trained on one domain (script identification) is adapted to perform well on a different but related domain (Character recognition). To implement the parallel processing approaches, the potential methodologies are data parallelism and model parallelism as all the models are trained using GPUs. Further to implement parallel processing pipeline parallelism can be used for composite character recognition which divides the training process into stages, where each stage processes a subset of the data and passes intermediate results to the next stage. As the characters in Gurumukhi script have modifiers present in the upper and lower zones, segmentation becomes more difficult. Hence, some kind of composite CR is needed to maintain the accuracy of CR. Dataset from the academic domain is required to design the text recognition system for this domain.

The success of proposed solution will be measured using various evaluation metrics including accuracy, precision, recall, F1 measure, Computation time, detection rate, recognition accuracy for segmentation, script identification and character recognition using the benchmarks datasets like IAM for Engligh documents etc. Keeping in view all the problems as highlighted above, there is a dire need to review the approaches used at each phase of text recognition for Indic and non Indic scripts in monolingual and multilingual texts. Future research can be focus on designing an efficient bilingual handwritten text recognition system with Indic and non Indic scripts. Further, to address the users of a particular public domain like academic, the designed bilingual text recognition system will be trained corresponding to the dataset. The research is focused on experimenting the latest approaches related to deep learning and ensemble of machine learning for handwritten text recognition in bilingual documents.

## 1.8 Objectives

The objectives of this research work are as follows:

1. To study and analyze the work done in the area of monolingual as well as bilingual recognition for Gurumukhi and English scripts including alphanumeric and special characters.

2. To propose domain based bilingual handwriting recognition for Gurumukhi-English script including alphanumeric and special characters.

3. To implement the proposed system in a usable form like a portal, an application, a database or a corpus etc. for academic domain.

4. To test and validate the usability of the proposed system.

## 1.9 Contributions of the Thesis

In this research work, an attempt has been made to design a domain based bilingual handwritten recognition system for Gurumukhi-English script. A bilingual handwritten text recognition system has various phases like data collection, pre-processing, segmentation, feature extraction, character recognition, etc. This thesis work covers each phase of text recognition. The contribution of this work is explained as hereunder:

1. A dataset from the academic domain has been collected to design a domain specific bilingual handwritten text recognition system, A dataset from the academic domain has been collected. The handwritten documents containing texts in English and Punjabi languages with a large variation in handwriting have been considered for the purpose of this research work. The contents of documents relate to the academic domain only. Further, for composite character recognition, training dataset of composite characters for Gurumukhi has been designed. The collected data has been pre-processed to bring uniformity in the dataset.

2. New algorithms have been developed and implemented for segmentation of handwritten text at the line, word and character levels. The experiments have been conducted on the collected dataset of documents from the academic domain.

3. The traditional feature extraction approaches have been used with deep learning based features. Different combinations of features have been evaluated for script identification of characters.

4. Script identification problem has been considered as a 5 class classification problem. The pre-segmented handwritten characters from English and Gurumukhi have been considered for the experimentation work. Further, to accelerate the process, transfer learning has been used to classify the text into different scripts.

5. Both the ensemble machine learning based approaches and traditional machine learning approaches have been used for classification of training data.

6. Composite character recognition has been performed using a composite character dataset for Gurumukhi 'aksharas'. A two level classification architecture has been designed to classify the data with a large number of classes. Transfer learning based pre-trained architectures like VGG19, and ResNet50 have led to improve the classification accuracy.

7. All the proposed algorithms have been tested and evaluated on the academic domain dataset. A GUI corresponding to each sub process of text recognition has been prepared.

8. The proposed algorithms for segmentation are computationally less expensive as it uses simple computer vision approaches without any need of training data or model to segment the text. Further single algorithm is sufficient to segment the bilingual text with curved, skewed or straight text.

9. For script identification, proposed algorithms are performing at character level and uses the transfer learning to utilize the power of deep learning for regional languages.

For character recognition our algorithm is able to recognize the 365 classes using pipeline parallelism with the help of transfer learning.

10. Our proposed algorithm for segmentation of bilingual text for Gurumukhi English documents has line improvement of 12%, and word of 11%. For script identification accuracy of 98% has been achieved as compared to traditional methods which was 94% using HOG. Although the time difference is huge. With the help of ensemble approaches accuracy for traditional classifiers and feature extraction methods have been improved. As it reaches 96% in 2638 seconds using voting based ensemble which is more efficient in terms of time as compared to deep learning methods.

## 1.10 Organization of thesis

The rest of the thesis is structured as follows. Chapter 2 reviews the various techniques of monolingual as well as multilingual text recognition for Indic and non Indic scripts. As text recognition has various phases, the complete literature review has been conducted corresponding to each phase. The latest research issues present for each phase of text recognition have been highlighted. The role of deep learning, transfer learning, machine learning and ensemble machine learning approaches for classification and feature extraction has been discussed in detail. The available benchmark datasets for monolingual as well as multilingual text recognition has been summarized. For feature extraction, traditional and deep learning based approaches are discussed. In chapter 3, the proposed framework has been presented in detail. It consists of the complete architecture of proposed BHTRforAD along with layer view in detail. The detailed architecture is presented for data collection and pre processing, segmentation, script identification, character recognition and to generate a doc file from recognized characters.

Chapter 4 discusses the various design and implementation details including algorithms and techniques. UML diagrams have been used to present the design details. It consists of structured and behavior modeling of the proposed framework. In structural modeling, the class and component diagrams have been designed to present the static view of the framework. Further to capture the dynamic view using behaviors modeling number of UML diagrams

are considered like Usecase diagram, Sequence diagram, State chart diagram, Collaboration diagram and Deployement diagram. This chapter also includes the implementation details like experimental setups, datasets, algorithm designed, techniques proposed etc. It discusses the algorithms designed for line, word and character segmentation. Further for script identification, the proposed approach has been described. Three OCRs designed corresponding to scripts are discussed in detail.

The chapter 5 named test and result analysis is design to present the testing results of proposed approaches. It consists the results of individual approaches as well as the testing results of complete proposed framework. The results have been found for each proposed approach corresponding to each phase. To evaluate the proposed approaches benchmark datasets like IAM, MNIST etc. have been used. To test the proposed framework, text cases have been designed and executed.The screenshots of the results of various text cases have been included in the Chapter 5.

Finally Chapter 6 concludes the thesis and points out the scope of further research.

# Chapter 2

# Literature Review

This chapter reviews the previous studies conducted in the field of text recognition. A modest attempt has been made to understand not only the basic concepts and issues of the present study, but also to find the research gap in the area under investigation. This work combines the research work of monolingual as well as multilingual recognition.

## 2.1 Monolingual and Bilingual HTR

This section provides an in depth analysis of the work done in the area of monolingual as well as bilingual text recognition.

### 2.1.1 Data Collection

Data collection is the most time consuming sub process in text recognition system. A good number of benchmark datasets are available for non-Indic scripts of monolingual text recognition. Among the non Indic scripts, Latin script has a wide variety of corpora like UNIPEN, CEDAR, IAM, IRONOFF, etc. UNIPEN provides a dataset for data collection of online handwritten text recognition with a wide variety of annotation hierarchy. Similarly, for online recognition, there is another dataset, i.e., UJIpenchars which includes 11000 samples [48, 49]. Further IRONOFF is the dual dataset providing data for online as well as offline text recognition. It includes the text written by French writers as words, characters and digits [50]. Another corpus for online text recognition is CEDAR. It is helpful for word seg-

mentation and recognition as it includes names, addresses, sentences, text lines and special characters, etc. [51]. For offline text recognition, IAM is the most popular corpus. It has paragraphs, words, text lines and characters dataset for text recognition [52]. For multilingual text recognition, IBM-UB is the corpus for online as well as offline recognition of text containing English and French languages [53].

Chinese script has a large character set having around 6000 characters. For Chinese character recognition, CASIA-OLHWDB and SCUT-COUCH are the popular datasets. CASIA includes various character categories, English letters, digits, etc. with various rotations [54]. The other SCUT dataset is a large vocabulary corpus including many other different datasets covering all the traditional, frequently and daily used symbols and words [55]. Similarly, for Japanese script, Kuchibue-d and Nakayosi-t are the popular datasets including many categories of Kanji, Hiragana, and KataKana [9].

Another widely used script after Latin is Arabic which is used by many languages such as Urdu, Farsi, and Arabic. For Arabic script, there are some popular datasets like QUWI, ADAB, LMCA, Latters, KHATT, and MADCAT. ADAB is a online handwritten word dataset which includes names of Tunisian towns and villages. Each word in dataset is prelabeled using sequence of numeric character references [56]. Another Arabic dataset, i.e., ARABASE is a dual purpose dataset providing corpus for online as well as offline recognition. It is designed on the basis of city names, literal amounts and digits [57]. Similarly, Research Group on Intelligent Machines has designed another dual purpose dataset, viz LMCA [58]. For multilingual text recognition, in Arabic script, there is QUWI dataset consisting of words from Arabic as well as English [59].

For Indic scripts, the research in text recognition is still at the infancy stage. Hence the number of available datasets for Indic scripts are quite less as compared to non-Indic scripts. A complete survey on various datasets for Indic scripts has been conducted by Hussain et al. for offline text recognition [60]. For online text recognition, Nethravathi et al. [61] and Singh et al. [62] have designed corpora for Tamil, Gurumukhi and Kannada scripts. For Gurumukhi script, the corpora is designed for word recognition considering application of map navigation, while for Kannada and Tamil, a wide variety of words, punctuation marks and symbols have been used. For multilingual text recognition, another corpora named PHDIndic_11 has

been designed for script identification [63]. It includes the page level dataset from 11 Indic scripts. This dataset has various applications for word spotting, writer identification, document image analysis and script identification, etc. Some of the researchers have designed their own multilingual datasets for script identification such as Guha et al. [64] [65]. Table 2.1 highlights the available benchmark datasets corresponding to scripts.

**Table 2.1:** Benchmark datasets available for various scripts.

| Dataset | Script | Text mode | References | Volume |
|---|---|---|---|---|
| KHATT | Arabic | Offline | [66] | 4,000 paragraphs |
| MADCAT | Arabic | Offline | [67] | 40k handwritten pages |
| AHTID/MW | Arabic | Offline | [68] | 3,710 text line images |
| IFN/ENIT | Arabic | Offline | [69] | 32,492 images |
| CMATERdb | Bangla and English | Offline | [70] | 18,000 words |
| IAM | English | Offline | [52] | 82,227 words |
| RIMES | French | Offline | [71] | 12,723 pages |
| MNIST | Digits | Offline | [72] | 70,000 images |
| HODA | Persian | Offline | [73] | more than 80,000 digits |
| MLT | multilingual | Offline | [74] | 20,000 images |
| Chars74k | English and Kannada | Offline | [75] | 74,000 images |
| IRONOF | French and English | Online and offline | [76] | 1,000 documents |
| TAMIZHI | Tamil | Offline | [77] | 1 lakh and 92000 |
| UNIPEN | English | Online | [78] | 5 millon characters |
| CASIAOLH-DB | chinese | Online and offline | [54] | 3.9 millon samples |
| QUWI | Arabic and English | Offline | [59] | 5,085 documents |
| PHDIndic_11 | 11 Indic scripts | Offline | [79] | 1,458 documents |

### 2.1.2 Pre-processing

There are some unavoidable variations in the data acquisition process of text recognition like noise, skew and slant etc. Thus, in order to remove such unwanted variations, pre-processing step is mandatory to preserve the actual shape of character. It includes many sub tasks like line removal, slant and skew correction, normalization, smoothing, and base line detection, etc.

Due to certain mechanical issues during the scanning of documents, some unwanted data enter the scanned image. It is named as noise emerged in image. There are many causes for such noise like dust and moisture present on paper, mechanical problem in scanner, etc. The early detection and removal of noise is mandatory for effective machine simulation of handwritten text as it can cause the problems like image degradation, destruction of prime features of image and inaccurate translation of image into ASCII characters, etc [80]. Further in document images, various types of noise are present such as salt and pepper, shadowing noise, white line dropout noise, and salient noise. Morphological operations are the widely used methods to remove some evident noises [81]. In morphological operations, the shape and size of connected components are analyzed to find the noise present. It is helpful in removing the noise present in handwritten words and characters. O'Gorman had designed the Kfill algorithm to remove the salt and pepper noise [82] .

Some documents like bank cheques, postal address forms, receipts and payment slips, etc. contain horizontal lines for the alignment of text written along horizontal axis. These horizontal lines get overlapped with handwritten text and make difficult to recognize the text. Hence, the detection and removal of such lines is a big problem in text recognition [16]. Dilation and erosion have been used by some researchers to remove such lines using threshold based techniques. However, the dilation operation distorts the shape of character which is unrecoverable even after the erosion operation. To avoid such loss, Yoo et al. have considered various junction points between the character and line crossing it. After line removal, only these junction points are restored rather than the whole character [83]. Similarly, block adjacency graph, horizontal projection profiles, and horizontal black pixel runs etc. have been used for line removal [84, 85].

Another crucial step in pre-processing of text images is reference line detection. It further helps in detecting the core region, height of word, and feature extraction of character sets etc. [80]. The most common approach for reference line detection is horizontal density histogram which uses horizontal densities [17]. Vinciarelli et al. have used threshold based Otsu methods to distinguish between reference lines and other lines [14].

Core region detected using reference lines serves as the prerequisite for skew correction. Cote et al. have used histograms in vertical direction to calculate the slope angle [86]. Similarly, Winger Ville distributions have been employed by Kavallieratou et al. to find the horizontal projection histograms [87]. Winger Ville distribution with maximum intensity is selected for slope angle. Morita et al. have used morphological operations to find the pseudo convex hull image. It results in minimal points which are used to draw the reference line [88]. Most of the approaches for skew correction draw the reference line by fitting a straight line through some points calculated based on projection profiles. These approaches are not applicable for text with noise, and many acsenders and decenders create problem to draw a reference line. Blumenstein et al. have divided the word image into two equal parts and founds the slope angle for each part individually [85]. To get efficient results, they have eliminated the ascenders and decenders of word which needs high computation cost. Rehman et al. considered structural features of text to find the slope angle [16]. Vinciarelli et al. have used cost function to detect the presence of slope in the text [14]. For slant correction, vertical strokes have been used to calculate the projection profiles. Many recognition models have used a fixed size of input, for which the scaling of text written is performed. Table 2.2 presents the state-of-the-art preprocessing techniques used by researchers.

**Table 2.2:** Pre-processing approaches for text recognition.

| Type | Script | Text mode | Methodology | References |
|---|---|---|---|---|
| Baseline detection | Arabic | Offline | Horizontal pixel density | [89] |

Continued on next page

**Table 2.2 – continued from previous page**

| Type | Script | Text mode | Methodology | References |
|---|---|---|---|---|
| Reference line detection | Arabic | Offline | 40k handwritten pages | [67] |
| Skew and slant correction | Latin | Offline | Projection profiles, Mathematical formulation, Connected component analysis, principle component analysis | [12] |
| Normalization | Script independent | Online | Size normalization | [90, 91, 92] |
| Line removal | Latin | Offline | Morphological shape analysis, Mathematical morphology, Foreground pixels analysis | [12, 85, 93] |
| Binarization | Script independent | Offline | Thresholding | [94, 95, 96] |

### 2.1.3 Segmentation

Segmentation is a crucial process in any text recognition system. It is performed at four major components of the document i.e. page level, word, line, and character level. A good number of algorithms are available for page and line level segmentation independent to script. Eskenazi et al., have performed a survey on segmentation algorithms for document images [97]. For character level, Saba et al., have presented a complete survey on the available algorithms for segmentation of touching characters [98]. Similarly, another survey has been conducted by Ribas for segmentation of handwritten digits [99]. The available segmentation techniques

are divided into three categories. The research work done corresponding to each category is explained as below:

1. ***Dissection based Approaches*** : The classical approaches consider projection profiles, bounding box analysis, and connected component analysis for segmentation of text. Projection profiles are the highly used approaches for segmentation of text lines. Sansam et al., have used the horizontal projection profiles along with gap trailing algorithm for segmentation of text lines in unconstrained handwritten text of Meietei Mayek. The proposed algorithm is able to segment the text with curved, skewed and closed lines. The gap trailing algorithm detects the mid-point between the two lines based on the projection profiles. Line and word segmentation goes simultaneously using vertical projection profiles for word segmentation [100]. The other work proposed by Vuvckovic et al. has also used the projection profiles to segment machine typed and printed characters along with spatial features of characters [101]. Projection profile methods are threshold dependent which vary corresponding to script, text area, type of text, etc. Hence, Ptak et al., have used variable threshold based on density distributions calculated using horizontal projection profiles [102]. Water flow based approaches are the most common approaches for text segmentation. Basu et al., have used water flow methods to segment text written in Bengali [103]. Morphological operations are also used along with projection profiles for text line extraction in handwritten documents. It helps in extracting the candidate features of text line which helps in finding the required text area for segmentation by histograms [104]. Jindal et al., have performed the segmentation of horizontally overlapped printed text. The proposed algorithm divides the text lines into horizontal stripes. The stripes with smaller size than defined threshold are merged to reconstruct the broken characters [105].

2. ***Recognition based Approaches***: These approaches use classifier to segment text with the help of features extracted from text. Susan et al., have used template matching for the segmentation of text area. Further, using adaptive threshold for image enhancement, 11 different kinds of texture features have bee computed. The distance between

texture and template features is used to separate the textual and non textual data. The researchers have considered five different kinds of datasets to evaluate the proposed approach [106]. Further, Sun et al., have used contour information with template matching to segment the printed multi-font Arabic text. They have used Cany edge detector to extract the contours of words and sub words. Template matching based on Freemain chain coding has been used to segment the descender characters [107]. Ruy et al., have considered the inter and intra word gap as a binary quadratic assignment problem. The connected components have been used to find the super pixel representations. Further, SVM classier has been used to perform structure learning, [108]. Hough transformations are also used for segmentation of text lines with the help of some post-processing rules. Further, to segment the words, Gaussian mixture model have been used which measure the inter and intra word gap [109, 110]. Similarly, to segment the curved and closed text lines, density estimation of each pixel have been considered [111]. Sharma et al., considered various features of text like header line, aspect ratio, etc. to solve the problem of under and over segmentation of Gurumukhi text [112]. Jo et al., have used CNN for the segmentation of mixed documents containing handwritten and printed text [113].

3. ***Holistic Approaches*:** These approaches are the combination of classical and recognition based approaches. Olszewska et al., have used contour extraction method and template matching method for segmentation. In first stage the active contours have been used for character extraction. Further, these extracted characters are recognized using template matching method to validate the segmentation of characters [114]. Similarly, Jung et al., have combined the heauristic and holistic methods to segment the touching characters [115].

Table 2.3 presents the comparative analysis of various segmentation approaches used in text recognition.

**Table 2.3:** A comparative analysis of various segmentation approaches in text recognition.

| Script | Techniques | Text Component | Dataset | Accuracy | References |
|---|---|---|---|---|---|
| Latin | Geometrical features | Character | IAM , CEDAR | 88.08% | [116] |
| Digits | Self organizing Maps | Character | NIST | 65.79% | [117] |
| Latin and Indian Languages | Binary quadratic and SVM | Word | ICDAR 2019 | 96.48% | [108] |
| Arabic | Bounding box analysis | Character | 450 word images | 87.00% | [118] |
| Latin | Projection profiles with variable threshold | Line, word, character | Nikola Tesla's documents | 80.58%, 77.57%, 86.14% for line, word and character respectively | [101] |
| Latin | Fuzzy inference system | Character | 57,293 cursive character images | 81.10% | [119] |
| Latin | RNN based LSTM | Word | ICDAR, SVT | 90.0% | [120] |
| Bangla | SVM and structural features | Character | 35,700 word images | 79.06% | [121] |
| Devanagari | Structural and statistical features | Characters | 18 documents | 85% | [11] |
| Chinese | Genetic algorithms | Characters | 428 character images | 88.90% | [122] |

Continued on next page

**Table 2.3 – continued from previous page**

| Script | Techniques | Text Component | Dataset | Accuracy | References |
|---|---|---|---|---|---|
| Japanese | Projection analysis | Characters | 456 character images | 83.0% | [123] |
| English and Greek | LSTM | Line | 9500 text line images | 98.19% | [124] |
| Indus and English scripts | Line | Region growing methods | 700 document images | 96.1% | [125] |
| Chinese and English | Gaussian mixture model | Character | 220 text images | 93.56% | [111] |

### 2.1.4 Feature Extraction

Initially, there were only two major categories of feature extraction approaches available for any text recognition task, i.e., statistical and structural features. These are also called the handcrafted feature extraction methods. The invention of deep neural networks has introduced one more category, i.e., non handcrafted features or deep learning features. This section performs an in-depth analysis of the work done in the field of feature extraction in text recognition for traditional as well as deep learning based features. Table 2.4 presents the comparative analysis of various feature extraction approaches corresponding for text recognition. The work done under various categories of features for text recognition has been explained as below:

1. ***Statistical features:*** Statistical methods of feature extraction include zoning, Zernike moments, geometric moments, Fourier descriptors, unitary image transformer, projection profiles, crossing and distance etc. based approaches. Further, gradient, wavelet, pixel distance, and convex hull based features are also identified to design sophisticated feature set for complex patterns. Kacem et al., have proposed texture based features for script identification. A run length matrix is defined as the number of runs having pixels of gray level *i* and run length *j* for image *I(i,j)*. A number of texture features can be extracted using run length matrix. They have used black run length matrix to design the features. Four variants of Black Run Length (BRL) features have been considered, i.e., simple BRL, restricted BRL, BRL statistics and combination of restricted and statistics [126]. Similarly, Brodic et al., have explored the texture features for script identification in Germen historical documents [127]. Another form of texture features, i.e., Non Sub Samples Contourlet Transform (NSCT) are used for script identification. NSCT has the properties of shift in variance and similar size of sub image and original image which help in preserving the original image information [128]. The transformation based features are rotation invariant. It includes Gabor filters, gradients, GLCM, wavelet, etc. Wavelet, curvlet, and ridgelet transformation have been used in feature extraction for multilingual recognition. Simple wavelet transforms are unable to fetch the edge representations of characters. Hence, multi direction transformations have

been used [129]. Gabor features are based on Gabor filters which are the sinusodial functions. These functions capture the features of image with various orientations and frequencies. Gradient based features convert the pixel representation of an image to gradient based representation. Hence, they are invariant to local geometric transformations. Two different kinds of Gradient based features have been computed for character recognition of offline Gurumukhi, i.e., decomposed gradient and non decomposed gradient vector [130]. Jamil et al., have used GLCM based features for multilingual text extraction from video images. Contrast, energy, correlation, homogenity and entropy have been calculated for each GLCM. These features are used to differentiate between the textured and non textured area. Further for script identification texture features like local binary patterns have been used [131]. Gabor filter based Modified log Gabor filters have been employed by Singh et al., to evaluate the dataset designed for bilingual text recognition[132]. Rani et al., have also used Gabor and Gradient based feature for script identification in Gurumukhi text [133]. Further, Kumar et al., have used zoning, horizontal peak distance and diagonal for multilingual script identification [134]. Cosine and wavelet transformation have been used by Pandey et al., for script identification in Hindi, Urdu and Gurumukhi [135].

2. ***Structural features:*** Structural features include crossing points, edges, lines, loops, aspect ratio, branch points, contours, and curves, etc. Such features use the geometrical attributes of smaller components to describe the pattern. Verma et al., have considered the contour information along with other properties of text like ascenders and descenders count, initial and ending points for feature extraction [136]. Jaeger et al., have extracted many structural features like curvature, vertical position, writing direction, pen up and down, slope, ascender, desender etc. for online text recognition [9]. Kumar et al., have designed two new structural features, i.e., parabola curve fitting and power curve fitting based for Gurumukhi offline text recognition. These methods divide the image into equal number of zones, and fit a parabola and power curve using least square methods [137]. Sahare et al., have proposed three new geometrical shape based structural features for segmentation of multilingual document images [138]. Water reservoir based structural features are widely used for the segmentation of text written

in document images. Roy et al., have considered two types of reservoirs for extraction of date field in multilingual documents. These features are helpful in extracting the background and foreground information of pixels[139].

3. ***Combination of features:*** Das et al., have explored the combination of two well known statistical and topological features, i.e., PCA and Quad tree longest run (QTLR) features. QTLR sometimes, fails to detect the global statistical information. Hence, this combination of PCA-QTLR efficiently recognizes the handwritten numerals [140]. Similarly, Khandju et al., have proposed a hybrid feature extraction approach for text recognition in Devanagari script. They have extracted the structural features of text like loops, number of end points, and intersection points. Further, the character image has been divided into zones and a quadratic curve model has been fitted on each zone to construct a feature vector [141]. Shivakumara et al., have combined Gradient based spatial and structural features for script identification in video frames[142]. Singh et al., have designed a heterogeneous classifier by combining two types of features, i.e., shape based and texture based features. The features extracted are HOG, elliptical based, and Gabor filters for script identification. Kumar et al, have combined structural and statistical features for text recognition in Gurumukhi [143].

4. ***Deep learning features:*** Du et al. have used deep learning based CNN features for writer adaptation in online Chinese text recognition. Further, a prototype based classifier has been used to classify the deep features [144]. Sarkhel et al., have modified the CNN architecture for recognition of isolated handwritten characters. A multi column and multi scale CNN has been designed. It uses individually three columns and three levels for feature extraction. The final output is the combination of all the columns. Different CNN architectures have been used at each level of each column [145]. Bhunia et al., have used CNN LSTM to employ the attention mechanism for script identification. It has resulted in the most relevant features of an image using attention mechanism. CNN has been used for the local feature extraction using most significant weights, while global features have been extracted using last layer of LSTM [39].

**Table 2.4:** A comparative analysis of various feature extraction methods employed in text recognition.

| Script | References | Application | Category | Feature set | Dataset | Accuracy |
|---|---|---|---|---|---|---|
| Hindi and English | [146] | Segmentation | Deep learning | Feed forward neural network | 68,000 characters | 99.78% |
| Latin and Bangla | [147] | Postal images recognition | Structural | Water reservoir | 7500 postal images | 93.14% and 86.44% for digit and character respectively |
| English, Chinese and Bengali | [148] | Scene text recognition | Statistical | HOG | ICDAR 2003, SVT, IIIT 5K datasets | 81.70%, 92.20%, 71% for ICDAR, Bengali, and Chinese respectively |

Continued on next page

**Table 2.4 – continued from previous page**

| Script | References | Application | Category | Feature set | Dataset | Accuracy |
|---|---|---|---|---|---|---|
| 11 Indic and non Indic scripts | Pati and Ramakrinana | Script recognition | Statisical | Gabor and DCT | 20,000 words | 94.80% |
| Arabic, Roman and Tamil | [149] | Script identification | Structural | Average stroke length, horizontal and vertical inter stroke direction, stroke density | 39 documents | 93.30% |
| English, Kannad, Tamil, Telgu, Gujrati and Hindi | [125] | Script identification | Structural | Connected components and dominant points | 700 scanned documents | 96.10% |
| Arabic, Urdu and Farsi | [150] | Machine printed text recognition | Statistical and Structural | Connected component and PCA | 189 documents | 100% |

Continued on next page

**Table 2.4 – continued from previous page**

| Script | References | Application | Category | Feature set | Dataset | Accuracy |
|---|---|---|---|---|---|---|
| Arabic and English | [151] | Language identification | Statistical | Projection profiles and run length histograms | 3176 text lines | 99.70% and 96.80% for text lines and word level respectively |
| English and Greek | [124] | Script identification | Deep learning | LSTM | 2,89,013 characters | 98.18% |
| 13 Indic and non Indic scripts | [152] | Script identification | Deep learning | DiscCNN | SIW-13 | 89.0% |
| Urdu and Latin | [38] | Cursive and non cursive script recognition | Deep learning | BLSTM | UCOM and UNLV-ISRI | 99.17% and 88.94% for Latin and Urdu respectively |

Continued on next page

**Table 2.4 – continued from previous page**

| Script | References | Application | Category | Feature set | Dataset | Accuracy |
|---|---|---|---|---|---|---|
| English, Devanagari and Marathi | [153] | Aged Indian documents recognition | Geometrical features | Perpendicular distance, triangular area, crossing count | 280 documents | 98.50% |
| 22 scripts and 97 languages | [154] | Online text recognition | Structural features | Water reservoir and stroke directions | UNIPEN and IAM-DB | error rate of .8% and 4.3% on UNIPEN and IAM-DB respectively |
| Chinese, Japanese, Korean, English | [155] | Script identification | Invariant features | Steerable Gabor filters | 575 sample images | 98.50% |

Continued on next page

**Table 2.4 – continued from previous page**

| Script | References | Application | Category | Feature set | Dataset | Accuracy |
|---|---|---|---|---|---|---|
| English, Kannada, Bangla, Oriya, Devanagri and Persian | [156] | Baseline detection in multilingual text | Statistical | Contour points | 5000 single and multi oriented text lines | Error rate 6.4% and14.8% for single and multi oriented text respectively |

Further, Ahmed et al., have used BLSTM based features for sequence learning in cursive handwritten texts [38]. For context modeling in text recognition, Chherawala et al., have considered BLSTM based features to train the context at feature level. Two traditional features, i.e., direction distribution and distribution have been employed along with BLSTM to provide context information [157].

### 2.1.5 Script Identification

Script identification approaches are categorized into two categories: appearance based approaches and structure based approaches. The selection of script identification method under these approaches highly depends upon the text component of document considered for script identification as script identification is performed at various levels like page, paragraph, line, word and character level. The first extensive research in script identification was made in 1994 by Spitz [158]. After its success, Ghosh and Ubul conducted two comprehensive surveys for script identification [7, 8]. Most of the research in script identification was devoted to Indic scripts due to the presence of a large number of scripts and languages in India. Therefore, a survey for script identification in Indian scripts was done by [159]. The various script identification methods proposed by researcher have been discussed as follows:

1. ***Script identification in Handwritten documents:*** Script identification in handwritten documents is more challenging than printed one because the writing style varies largely with age, mood and gender of the writers. It creates a large number of shapes for a single shape of character which creates a problem for similar shaped characters. Hochberge et al., have considered the various features of document like standard deviation, mean, aspect ratio, number of holes, and vertical centroid of connected components for script identification. These features are classified using Fisher Linear Discriminators classifier [**?** ]. Another work of script identification at text block level is done by Ghosh et al., using probabilistic approach for cluster analysis [160]. Water reservoir and fractal based features along with neural networks are highly used approaches for script identification of Bangla, English, Roman and Devanagari in Indian post documents at word level [161, 162, 163]. Fractal features use fractal signature

which is calculated from the area mapped by gray level function to the document image. Many other researchers like Roy et al have also considered fractal dimension of text for script identification at word level in Indic scripts [164, 165]. Dhandra et al., have used two step process for script identification of numerals and words. Local and global features have been used to identify the words of Roman, Kannada, Devanagari scripts [166]. Pal et al., have explored script identification at the character level for numerals of six Indic scripts using modified quadratic classifier [167]. Razzak et al., have used Fuzzy rules to identify the numerals of Urdu and Arabic script [168].

2. ***Script identification in Printed documents:*** A good amount of studies have been conducted for script identification in printed documents. For printed documents, most of the work has been done for page level script identification. Spitz et al., have done a great work for script identification at the text line level using character optical density, vertical distributions of upward concavities in characters [169, 170]. Further, the researchers have used clustering for script identification of 26 languages[158]. Projection profiles have been used To classify the languages. Ding et al., have used statistical features for script classification of European and Oriental scripts [171]. Gabor filters and GLCMs have also been used by [172]. To reduce the computational cost of Gabor filters, steerable gabor filters have been considered for script identification [155]. Similarly, Waked et al., have used bounding box analysis, character density distributions and horizontal projections. The results have proved that statistical features are more robust as compared to structural features [173]. Hochberg et al., have used template based clustering for script identification [174]. They have designed the template for textual symbols extracted from documents. Further, Hamming distance has been used between the template and textual symbols. Morphological operations are also used for feature extraction of scripts using erosion dilation along with nearest neighbor classifier. Chaudhury et al., have used Fourier transformation along with horizontal projection profiles for script identification of text blocks [175]. FFNNs have been employed along with morphological operations for script identification of Latin, Devanagari and Kannada scripts [176, 177]. Pal and Chaudhury have considered script specific characteristics like presence of principal strokes, characters having more than four vertical

runs, water reservoir based features in text lines [178]. Script identification at the word level is quite challenging due to the limited area of information available. Lee et al., have used Self organizing maps which identify the script of each individual character [179]. Ablavsky et al., have used connected components analysis based on geometric properties like compactness for shape description and Cartesian moments [180]. For Indic scripts, the bilingual recognition system designed by Jawahar et al., detects the presence and absence of shirorekha for script identification [181]. To evaluate the results, further contextual information has been used. The results of script identification at the character level can be combined for word level script identification using Viterbi algorithm [182].

3. ***Script identification in hybrid documents:*** Hybrid documents consist of printed and handwritten text. The major applications of such processes are bank cheques, postal form, passport forms etc. Moussa et al., have proposed an approach for script identification in hybrid documents containing Arabic and Latin texts [183]. Kanoun et al., have used morphological operations and geometric analysis to differentiate the scripts at text block, line and connected component level in hybrid documents [184]. Benjelil et al., have considered steerable pyramid transform for script identification at word level [185]. Another work by Saidani et al., has considered the structural features to identify the scripts at the word level for Arabic and Latin [186] .

4. ***Script Identification in Video Frames and Camera based Images:*** Apart from document analysis, another recent trend for script identification is for text recognition in video frames. It helps in indexing and retrieval of images and videos. Text recognition in images and videos at international level has great significance for tourism applications. The recognition of such multilingual text needs script identification along with many sub processes like text detection, localization, segmentation and binarization. The researchers in their work have proposed an approach which first localizes the text in image or video and extracts the low level features like edge pixel density, energy and Cartesian moment of edge pixel etc [187]. Script recognition in video frames and scene images is difficult as it suffers from complex conditions like low resolution,

noise, orientation, complex background, orientation problems etc. Zhao et al., have used spatial gradient based features for script identification [188]. Phanet et al., have used two features called smoothness and cursiveness for script identification in video frames [189]. Sharma et al., have used Zernike moments and Gabor gradients along with SVM for script identification at the word level [190]. In yet another work texture features like GLCM, LBP etc. have been used which help in identifying the script for images with low resolution, noise and complex background [191].

5. ***Script identification in online text recognition:*** The advances in pen computing devices have raised the need for many online document recognition systems. In general, the approaches available for script identification in offline data are not applicable for online text. Hence, an online script recognizer is needed for online text recognizer. Lesser work has been found for online script identification as compared to offline script identification. Lee et al., have used HMM recognizer to recognize the entire character set consisting of characters from more than one scripts for online text. The basic characters of language and intermixed language characters are modeled by constructing the hierarchical HMMs. During recognition, Viterbi algorithm has been used to find the optimal path. This helps in finding any combination of basic language characters [192]. Another prominent feature used in online script recognition is the writing direction. Temporal sequence of strokes available in online writing is helpful to find the writing direction of text. As some scripts like Devanagari, Gurumukhi and Latin etc. use left to right direction while some other scripts like Arabic and Hebrew use writing style from right to left. Namboodri and Jain have proposed nine features for script identification in online text such as average stroke length, horizontal inter stroke direction, shirorekha confidence, stroke density, aspect ratio, reverse direction, average horizontal stroke direction, and average vertical stroke direction. Further, in their extended work, they have considered two more features like vertical inter stroke direction and variance of stroke length [193].

**Table 2.5:** A comparative analysis of state-of-the-art approaches for script identification.

| Script | References | Application | Text Component | Feature set | Classifier | Dataset | Accuracy |
|---|---|---|---|---|---|---|---|
| Hindi and English | [146] | Handwritten | Character | Pixel Plot and Trace based | FFNN | 68,000 characters | 99.78% |
| Latin-Bangla, Devanagri-Latin | [194] | Handwritten | Document | Gabor filter | 92.32%, 95.30% and 93.78% | | |
| 12 Indic scripts | [195] | Handwritten | Word | Texture and shape based features | MLP | 7200 words | 95-98% |
| 13 scripts Indic and non iNdic | [152] | Natural scene images | Word | CNN | DiscCNN | SIW-13 | 89.00% |

Continued on next page

**Table 2.5 – continued from previous page**

| Script | References | Application | Text Component | Feature set | Classifier | Dataset | Accuracy |
|---|---|---|---|---|---|---|---|
| Indic and Non Indic scripts | [39] | Scene images and videos | Word | CNN | CNN-LSTM | SIW-13, CVSI2015, ICDAR-17 and MLe2e | 96.50%, 97.75%, 90.23%, 96.70% for SIW-13, CVSI2015, ICDAR-17 and MLe2e respectively |

Continued on next page

**Table 2.5 – continued from previous page**

| Script | References | Application | Text Component | Feature set | Classifier | Dataset | Accuracy |
|---|---|---|---|---|---|---|---|
| English, Chinese, and Bengali | [148] | Scene text | Character | HOG | SVM | 6182 Chinese, 19530 Bengali characters, ICDAR2013, SVT, IIIT 5K-words | 81.70%, 92.20%, 71.00% for ICDAR, Bengali, Chinese respectively |
| English and Greek | [124] | Printed text | Text line | Deep networks | LSTM | 9500 text lines | 98.18% |

Continued on next page

**Table 2.5 – continued from previous page**

| Script | References | Application | Text Component | Feature set | Classifier | Dataset | Accuracy |
|---|---|---|---|---|---|---|---|
| 11 scripts | [196] | Torn documents | Character and word | Zernike moments | SVM | 142 documents | 81.39% and 94.65% at character and word level respectively |
| Thai and English | [197] | Printed | Words | Water reservoir and topological features | SVM | 6110 words | 99.36% |
| 11 Indic and non Indic scripts | [198] | Printed | Words | Gabor and DCT | 20000 words | 94.80% | |

Continued on next page

**Table 2.5 – continued from previous page**

| Script | References | Application | Text Component | Feature set | Classifier | Dataset | Accuracy |
|---|---|---|---|---|---|---|---|
| Arabic, Roman and Latin | [149] | Online handwritten | Document | Stroke density, average stroke length, horizontal and vertical inter stroke direction | Information retrieval based | 39 documents | 93.30% |
| Gujrati, Hindi, Tamil, Kannada, Telgu and English | [125] | Printed | Text lines | Connected components and dominant points | Template based | 700 scanned document images | 96.10% |
| Arabic and Latin | [126] | Handwritten and printed | Word | Texture features | KNN | IAM, IFN-ENIT, AHTID/MW and APTI | 98.92% |

Continued on next page

**Table 2.5 – continued from previous page**

| Script | References | Application | Text Component | Feature set | Classifier | Dataset | Accuracy |
|---|---|---|---|---|---|---|---|
| English, Devanagari, and Bangla | [199] | Signature identification | Word | Zernike moments and HOG | SVM | 6300 samples | 92.14% |
| Bangla, Roman, Devanagari | [65] | Scene images | Word | EAST | CNN | IMDB, pinterest | 99.82% |
| 12 Indic scripts | [64] | Handwritten | Lines | MLG, HOG, DHT | KNN, MLP, SVM | 7200 text lines | improvement of 2-5% in accuracy |

Another work has used the Fuzzy Online Handwriting Description Language to describe the character features. It uses fuzzy values for each character in terms of fuzzy rules [200]. Table 2.5 presents the comparative analysis of state-of-the-art in script identification.

### 2.1.6 Classification

Classification is the crucial step for any machine learning based tasks. Based upon the features extracted from previous steps or using automatic feature extraction capabilities, classifier assigns the highly possible class to the data from set of classes. In this work, the classification techniques have been reviewed under two categories, viz. shallow architectures, and deep learning based architectures as follows:

1. ***Shallow architectures based classifiers:*** All the traditional machine learning based classifiers like HMM, SVM, MLP, and KNN etc. appear under shallow architectures. Such classifiers use single layer of non linear feature transformation to map raw input to the problem specific feature space. These methods require handcrafted features to perform the classification task. During the span of 2001 to 2010, HMMs were the widely adopted classifiers to model the input data for text recognition. The sequence modeling capability of HMM networks makes it suitable for the online text recognition task. Many researchers have adopted HMM for monolingual as well as multilingual text recognition. Apart from text recognition HMM has been used for keyword spotting in handwritten documents by Thomos et al. [201]. However Roy et al. have considered HMM for word recognition after zone identification during segmentation. They have recognized the middle zone characters after segmentation using Viterbi force. Pyramid HOG based features have been used for recognition [202]. Simple HMM models are efficient to recognize the one dimensional features. For multi dimensional features or bi dimensional signals the solution proposed by researchers is to combine the multiple classifiers. Jayech et al., have proposed multi stream HMM to model the statistical and structural features for segmentation free Arabic text recognition. These features are spread over the rows and columns, hence multidimensional HMM models are used [203]. Another variant of HMM models, i.e., Bernoulli HMM

is used for offline word recognition. It bypasses the feature extraction process by directly passing the columns of raw image pixels to Bernoulli HMMs [204]. Ghods et al., have used the fusion of semi continues HMM classifiers for online text recognition in Farsi [205]. SVM is the other widely used classifier from shallow classifiers for text recognition tasks. It uses various types of kernels based upon the kind of data used for classification. Sampath et al., have designed a multi kernel spherical SVM based on fuzzy. It uses fuzzy triangular membership function to design a multi kernel function [206]. Simistira et al., have used SVM for recognition of mathematical formulas written in online mode. They have considered probabilistic SVMs to represent the spatial relation between mathematical functions. Further, stochastic context free grammar has been adopted for the compliance of recognition results [207]. SVM with Polynomial kernel has been used for the recognition of Persian handwritten digits along with adaptive feature framing approach [208]. Niu et al., have combined CNN with SVM for recognition of handwritten digits. CNN performs automatic feature extraction while SVM is used for classification [209]. Similarly, Omar N. Al-Boeridi et al., have designed hybrid classifier using ANN and SVM for the recognition of Roman words [210]. Another simplest shallow architecture is KNN. The major modifications proposed by many researchers for KNN classifiers are based on the distance measure used in KNN. Prasad et al., have used weighted KNN classifier which uses $\chi^2$ distance measure for Gujarati character recognition [211]. Porwik et al., have used KNN method for signature recognition [212]. Yamashita et al., have proposed various kinds of distance measures in KNN like global affine transformation and global projection transformations based [213]. Sternby et al., have used template matching for the recognition of online handwritten text of Arabic script [30]. Template based systems are considered robust under conditions of limited training data. It is easy to handle the manual conflicts due to explicit modeling of reference structure or template. Bag et al., have proposed template matching based approach for compound character recognition of Bangla script. It decomposes the Bangla characters into appropriate stroke segments using some rules and further uses template matching to extract the convex shape primitives[214].

2. ***Deep architectures based classifiers:*** The strength of deep architectures lies in their capability of extracting features automatically. Roy et al., have performed the layer wise training of deep neural network for recognition of Bangla compound characters. To provide better training and faster convergence RMSprop optimizer has been used. The layer wise training uses the supervised learning concept which incrementally adds and trains convolution and pooling layers [36]. For text recognition in scene images, the segmentation of touching characters is a challenging task. Su et al., have used RNN based LSTM networks for segmentation free word recognition in scene images. Initially, it generates the sequential signals from word images and trains multiple RNNs using these signals [120]. Deep neural networks demand high computational cost and large number of hyper parameters to generate state-of-the-art results which restrict its usage for portable devices. Xiao et al., have solved this issue for Chinese handwritten character recognition by introducing two new techniques, i.e., Adaptive Drop-weight and Global supervised Low rank expansion. The proposed algorithm reduces the size of nine layer Deep CNN by $1 \div 18$ of the original size and computation cost by $1 \div 9$ [215]. Zhang et al., have used CNN for the discovery of similar Chinese characters in online handwritten text. It helps in identifying the miss classified character pairs [216]. Naz et al., have used multi dimensional RNN for script identification of Urdu-Nastaliq text. They have used sliding window for extraction of statistical features on lines of text and CTC based RNN for labeling the character sequence [217]. Further, Lai et al., have proposed Drop Distortion based training strategy to train DCNN for online Chinese character recognition. It helps in lowering the character distortion during training of CNN. Further, path signature is used for feature extraction along with spatial stochastic max pooling layer based on fractional max pooling [218].

3. ***Ensemble based machine learning classifiers:*** Ensemble of classifiers is designed by combining the complementary nature of different classifiers which eliminates the weakness of individual classifiers and considers their strength. There are mainly two categories of ensemble machine learning classifiers, i.e., classifier selection and classifier fusion. Classifier selection selects the best performing classifier from the set of classifiers. The best example of this is cross validation. On the other hand, classifier

fusion performs fusion on the combined output of individual classifier to result the final decision. The prime examples of this are Majority voting, Neural Network, and Dempster shafer theory, etc. Singh et al., have used ensemble classier with MLP as base classifiers for the recognition of handwritten Devanagari numerals. Feature selection has been performed using information theory measures, while Voting, Dempster shafer, Bayes, and decision template have been used for classifier combination [219]. Similarly, Singh et al., have used correlation based classifier combination [220]. Salimi et al., have proposed a new classifier combination technique named reliable multi phase PSO. Singular value decomposition based algorithms have been used to design the base classifiers. The significance of proposed approach lies in its independence to training set size [221]. Oliveira et al., have proposed feature selection based ensemble approach for text recognition. The method first selects the features and generates the classifiers corresponding to feature sets. After this, the best team of classifiers and features is selected. The proposed approach shows great improvement in error rate compared to classic methods of ensemble approaches like bagging and boosting [222]. Kessentini et al., have used Dempster shafer theory to design ensemble classifier for writer identification problem. Three base classifier based on SVM has been designed using three different features, i.e., edge hinge with fragment length 6 and 7, run length based features [223]. Elanwar et al. have used ensemble of classifiers to convert the raster images of documents containing text or graphics into machine accessible format. Ensemble of SVM based classifiers has outperformed in classifying the document image patches as compared to traditional methods like RLSA and RDI-CleverPage [224]. Dempster shafer based classifier combination strategy provides robust and accurate classifiers generated from incomplete, biased and imprecise information sources. Kessentini et al., have used DST for ensembling of HMM based classifiers for word recognition along with post processing module based on a set of various decision strategies. Further, to handle the rejected samples, multi stream HMM has been proposed [225]. De Stefano et al., have proposed Bayesian network based classifier combination approach for handwritten recognition. It considers ensemble of classifiers as a pattern recognition problem where each feature set corresponding to

input pattern uses classifier to output the class. Bayesian network uses the probability distribution for final classification [226]. Bertolami et al. have used bagging, language models and random feature space for ensemble creation for text line recognition. The final combination for the word sequences is generated by using the voting based strategy on individual classifier's results [227]. Masoudnia et al., have solved the signature verification problem using ensemble of CNNs based on multiple losses considered. The use of various loss function helps in learning the diverse representation of input pattern [228]. Li et al., have used boosting based ensemble for person re identification task [229].

4. ***Transfer learning based classifier:*** Pramanik et al., have considered segmentation free recognition of handwritten Bangla city names. They have used five traditional features with traditional classifiers for recognition. Further, five different deep architectures have been used for transfer learning. A comparison of transfer learning based and a seven layer FCNN has been used [46]. Similarly Bonyani et al., have used transfer learning based architectures for Persian digits recognition using ResNet50 and VGG16 [73]. Balah et al., have solved the Arabic handwritten character recognition using pre-trained CNN models i.e. VGG16, VGG19, MobileNetV2 [230].

Table 2.6 and 2.7 performs comparative analysis of various deep learning and traditional feature extraction approaches used in literature for text recognition.

**Table 2.6:** A comparative analysis of traditional classifiers for multilingual text recognition.

| Classifier | References | Text mode | Script | Feature set | Dataset | Accuracy |
|---|---|---|---|---|---|---|
| HMM | [147] | Offline | Bangla and English | Water reservoir-based | 10,342 handwritten words and 1250 printed words | 51.41% |
| SVM | [198] | Offline | Hindi, English (printed) | Gabour and Discrete Cosine Transform (DCT) | 20,000 words | 98% and 89% for bi-script and eleven script respectively |
| NN | [151] | Offline | Arabic, English | Horizontal projection profiles and run length histograms | 1,200 text lines for training data of Arabic and English, 1,976 text lines for testing | 99.70% and 96.80% for text line and word level respectively |
| SVM | [231] | Offline | Chinese, English, Japanese | 256 dimensional feature vector | 8,89,846 samples | 99.92% accuracy and 76.29 computing speed (Character/second) |

Continued on next page

**Table 2.6 – continued from previous page**

| Classifier | References | Text mode | Script | Feature set | Dataset | Accuracy |
|---|---|---|---|---|---|---|
| ANN | [154] | Online | 22 scripts and 97 languages | 23 geometric point-wise features with character-wise features | Varies from 10,000 to several millions | Lowest character error rate (CER) of 0.8% and 4.3% for UNIPEN and IAMDB respectively |
| LinSVM | [148] | Offline | English, Chinese, Bengali | Histograms of oriented gradient | 10,658, 15,250 and 3,796 character samples of Chinese, Bengali and English respectively | 81.70%, 92.20% and 71.20% for English, Bengali and Chinese respectively |
| Template Matching | [232] | Offline | 13 scripts | Connected components | 268 document images | Accuracy varies corresponding to script |
| K-NN | [150] | Offline | Arabic, Urdu, Farsi | Connected components with PCA | 375 documents | 100% |
| LDA | [233] | Offline | Six scripts with 8 languages | Mean, standard deviation and skew of 5 connected components | 496 documents | 88.00% |

Continued on next page

**Table 2.6 – continued from previous page**

| Classifier | References | Text mode | Script | Feature set | Dataset | Accuracy |
|---|---|---|---|---|---|---|
| SVM | [? ] | Offline | Bangla, Oriya, Kannada, English, Devanagari and Persian scripts | Orientation invariant features | 5,000 text lines | Maximum error rate of 14.80% |
| Hybrid | [234] | Offline | Arabic, Chinese, Hindi and Korean | Gabour filters | Not mentioned | Maximum accuracy of 98.08% for Hindi |
| Template matching | [125] | Offline | Indus scripts | Three feature vector from dominant points of Connected Components | 700 documents | 96.1% |
| HMM | [235] | Online | Hangul and English | Chain code-based | 52,000 Hangul syllables, 50,500 English alphabets and 4,100 digits | 83.82% |

Continued on next page

**Table 2.6 – continued from previous page**

| Classifier | References | Text mode | Script | Feature set | Dataset | Accuracy |
|---|---|---|---|---|---|---|
| Template matching | [236] | Offline | Six Indic and non-Indic scripts | Feature vector resulted from vertical component cuts | 153 training and 280 test documents | 91.56% for language identification and 97.29% for script identification |

**Table 2.7:** A comparative analysis of deep learning based classifiers for text recognition.

| References | Textmode | Script | Classifier | Dataset | Accuracy |
|---|---|---|---|---|---|
| [146] | Bilingual Offline | English and Hindi | FFNN | 68,000 samples | 99.78% |
| [38] | Offline | Urdu and Latin | BLSTM | UCOM and UNLV-ISRI | 99.17% for Latin and 88.94% for Urdu |
| [215] | Offline | Chinese | CNN | 26,78,424 samples | 2.90% error rate |
| [237] | Offline | Latin | BLSTM | IAM offline | Word error rate 16.01% |
| [43] | Offline handwritten | French | RNN | RIMES | 77.06% |
| [124] | Offline printed | English and Greek | LSTM | 2,89,013 characters | 98.50% |
| [152] | Offline | 13 scripts including Indic and non-Indic | DiscCNN | SIW-13 | 89.00% |

### 2.1.7 Post-processing

This section highlights the work done in the area of post processing for text recognition. There are mainly three types of errors for the results generated by recognition models. To solve the non word errors, n gram models based on Markov chains are mostly common used. The other type of errors are handled by using some linguistic data and statistical language models. Post processing phase provides a confidence measure to improve the recognition results of classifiers. There are mainly four key factors which affects the post processing results, i.e., lexicon size, searching strategy, statistical language models, and confidence measure. Li et al. have analyzed the seven language models based on perplexity measures. To measure the candidate confidence, many approaches like maximum posterior probability, logistic regression, and adaptive confidence transform have been used [42]. Sometime, the large lexicon size used in post processing results in increased recognition time. Carbonnel et al., have solved it by proposing a lexicon reduction approach and new edit distance to search the word in lexicon [238]. Farooq et al., have used topic categorization method to reduce the lexicon size in post processing [239].

Apart from traditional approaches the latest trend in post processing is the use of deep learning based language models. The sequence modeling capability of LSTM and BLSTM is highly useful in post processing the cursive and sequential text recognition. It results in Lexicon free post processing. Wang et al. have used RNN as sequence modeling approach for scene text recognition [240]. Naz et al. have used LSTM for labeling of character sequences[241].

Table 2.8 conducts comparative analysis of various post processing approaches used in text recognition.

**Table 2.8:** A comparative analysis of post processing approaches for text recognition

| References | Script | Text mode | Language Model | Classifier | Dataset | Results | Problem Resolved |
|---|---|---|---|---|---|---|---|
| [242] | Arabic | Offline | Statistical using posterior probability | SVM | 3,000 handwritten numerals | 99.30% | Miss classification of structurally similar characters |
| [243] | Arabic | Online | Word-part dictionaries, letter shape models using DTW | HMM | 800 training words and 2,358 testing words | 95.44% for writer independent on 40K dictionary size | Delayed strokes |
| [244] | Devanagari, Tamil | Online | Lexicon-based | Recurrent HMM | 20K lexicon size | 95.29% and 96.06% for Devanagari and Tamil (Top 5) respectively | Recurrent HMMs and Bag of symbol approach recognizes the words independent of symbol writing order |

Continued on next page

**Table 2.8 – continued from previous page**

| References | Script | Text mode | Language Model | Classifier | Dataset | Results | Problem Resolved |
|---|---|---|---|---|---|---|---|
| [245] | Chinese | Offline | Posterior probabilities with geometric and linguistic context | Maximum character accuracy | CASIA-HWDB of 7,356 character classes | 90.75% | Character segmentation-recognition framework for unconstrained HCCR |
| [246] | Tamil | Online | Bigram language models at symbol or character level with DTW | SVM | 15,000 handwritten isolated Tamil words | 93% at character level and 81.60% at word level | Ambiguity in confusing symbols is resolved using expert classifier based on DTW |
| [247] | Chinese | Online | Word Bigram Markov Model with three stages | Markov model | More than 2.5 GB with 1,140 million Chinese characters | 95% for top 10 candidates | Unrecognized errors resolved using 3 stage post-processing |
| [41] | English | Offline | Statistical language model | HMM | 45,800 character tokens | 90% | Generates confidence score using likelihood ratios |

Continued on next page

**Table 2.8 – continued from previous page**

| References | Script | Text mode | Language Model | Classifier | Dataset | Results | Problem Resolved |
|---|---|---|---|---|---|---|---|
| [248] | English | Offline | Topic based language models | Maximum entropy and Naive Bayes | IAM | 40.63% | Handles noisy documents |
| [42] | Chinese | Offline | Statistical language models at character, word and class level | Combination of 4 estimation methods | 66,000 characters | 97.65% | Combines 7 LMs, 4 confidence measures with different sizes of lexicons |
| [43] | French | Offline handwritten | N gram, string with similarity | RNN | RIMES | 77.06% | Increases the dictionary coverage by processing out of vocabulary words |
| [249] | Arabic | Online | Cross word trigrapheme HMM model and high order language model | HMM | ADAB | Maximum accuracy of 87.47% with lowest average time 0.15 sec | Handles delayed stroke problem |

Continued on next page

**Table 2.8 – continued from previous page**

| References | Script | Text mode | Language Model | Classifier | Dataset | Results | Problem Resolved |
|---|---|---|---|---|---|---|---|
| [250] | English | Offline printed | Language model with string similarity | HMM | 55,600 legal documents | Character error rate of 5% for high resolution images | Levenshtein string distance improves Viterbi-based search |
| [251] | Gurumukhi | Online | Set theory | Not mentioned | 27,231 samples of strokes | 80.30% for character formation | Computational complexities of character formation was modeled |
| [252] | Korean | Offline | Improved Levenshtein metric | Not mentioned | 1.3 million words | 83.60% recognition rate | Corrects the misrecognized text in signboard images |
| [253] | Chinese | Offline handwritten | Statistical language model | K- means clustering | 1,20,000 images | 81.20% | Recognition rate improved by 12% from previous work |
| | | | | Deep Architectures | | | |

Continued on next page

**Table 2.8 – continued from previous page**

| References | Script | Text mode | Language Model | Classifier | Dataset | Results | Problem Resolved |
|---|---|---|---|---|---|---|---|
| [44] | Latin | Offline | BLSTM | Hybrid HMM and NN | IAM offline | Word error rate 16.01% | Generalization capability of NN combined with HMM results into low error rate |
| [254] | Latin | Offline | RNN | CNN+RNN | Google street view | 90.20% | Recognizes both the lexicon-based and lexicon free text without segmentation |
| [217] | Urdu-Nastaliq | Offline | RNN | RNN with CTC | 10,000 text lines | 96.40% | CTC output layer labels the character sequences |

## 2.2 Challenges in Monolingual and Bilingual Handwritten text recognition

From the above sections,it seems that a lot of work has been done in text recognition. However, the efficient text recognition systems for regional especially Indic scripts is still lacking from the state-of-the-art work. Moreover there are many areas where text recognition is not explored yet. Some research issues have been identified during literature survey of handwritten text recognition at each phase. This section highlights the major issues to be resolved at each phase of handwritten text recognition.

### 2.2.1 Data collection

There are a good number of benchmark corpora available for monolingual text recognition while for multilingual text recognition there is a wide space of research. It is a challenging task to collect the dataset for multilingual text recognition with large variations of samples corresponding to each script character set. Some of the major issues related to dataset for test recognition are as follows:

1. There is an urgent need to developing the benchmark corpora for Indic scripts. Most of the corpora available are for Latin and Arabic scripts.

2. Data annotation process for data creation should be standardized in terms of writer's name, age, profession, and gender etc.

3. Each domain has a different category of users. Hence, domain specific datasets are more needed for real time tasks of CR. The collection of such ground truth data is really a challenging task.

4. For real world applications like forensic analysis and document analysis, the evaluation constraint of dataset should be designed based on their usage.

5. There is a need to provide a common platform for the evaluation of all the benchmark datasets.

### 2.2.2 Pre-processing

Pre processing step plays a vital role in preparing the dataset needed for subsequent phases of text recognition. It provides ready to use data for segmentation, feature extraction and classification phases. The major challenges faced during pre-processing of text are as follows:

1. The solution provided by many researchers for slant and skew correction is computationally expensive. The approaches used to find the error angle requires a lot of parameter optimization.

2. The pre-processing approaches based on projection profiles used for distinguishing between straight segments and scattered dots are not successful.

3. For many scripts which have header line present, pre-processing is a challenging task. As header line adds complexities due to its multipart nature and shape similarity with other strokes etc.

4. In online text recognition, some scripts have delayed strokes like in Arabic and Indic. It requires reordering of strokes in the pre-processing phase using some reference numbers or zone identification of strokes etc.

### 2.2.3 Segmentation

From the literature survey, it can be concluded that touching and horizontally overlapped text results in higher segmentation rates. There are number of issues which need to be tackled for accurate segmentation of text. These are explained as follows:

1. Lesser amount of work is found to segment the textual and non textual data in document images. Further more algorithms are needed for segmentation of online text.

2. There is a need to create corpora with diverse training data including various kinds of documents with curved, skewed and touching text. Because in review, a large difference has been found between the training and testing sets.

3. The evaluation criteria for proposed segmentation algorithms should be based on number of mixed, over and bad segmentation.

4. Sometimes the similar algorithm generates different results corresponding to the same corpus. Hence, there is a need for stable algorithms based on deep learning and statistical approaches.

5. The linguistic information should be considered to segment the touching data in multi sized and multi oriented text. The post processing based on linguistic information would be helpful in identifying the touching characters.

6. There is a need of some variable threshold based projection profile methods to segment the slanted and skewed text. The algorithm should be designed by keeping in mind the segmentation rate and algorithm complexity.

### 2.2.4 Feature Extraction

The traditional local and global features available during 90s are not efficient to distinguish between similar characters in multilingual text. Further, to recognize the text in complex patterns more efficient feature extraction approaches are needed . Deep learning is the new tradition for feature extraction which is successful also. To improve the feature extraction process, some issues which need to be resolved have been identified as below:

1. Feature vector should be constructed in such a way that it results in minimum within class variability and maximum between class variability.

2. Features identified should be independent to font and size of text, robust and rotation invariant.

3. To design a robust feature vector for multilingual text, some script specific information should be included. It includes compound characters, presence or absence of header

line and modifiers etc.

4. Some more approaches for feature selection and evaluation should be explored.

5. Ensemble of features using various combination of features should be used for multilingual text recognition.

6. Some domain specific features need to be extracted to design domain specific text recognition system.

### 2.2.5 Script identification

Many scripts have almost similar character set which makes script identification a difficult task. Deep learning based approaches have made it somewhat easier. Although there are still many unresolved issues are present such as:

1. In multilingual text, similarities and dissimilarities exist between classes and within classes. To capture such complex patterns a single feature set and classifier is not sufficient.

2. However, less work has been reported for script recognition in handwritten text for Indic scripts at character level. There are some words consisting of characters from different scripts. Hence, script identification at character level is needed.

3. Moreover, feature extraction at the word, page and line level is difficult and time consuming as compared to feature extraction at the character level.

### 2.2.6 Character recognition

There are number of classifiers present for classification tasks especially for text recognition. However this area requires consistent research efforts to make the classifiers accurate and efficient. From the literature we have found some challenges for classification or recognition of text such as:

1. From shallow architectures, template matching systems are noise sensitive while statistical methods like SVM, and HMM, etc. fails to identify the similar data between classes.

2. Template matching and DTW methods based on structural mapping are computationally expensive. The use of probabilistic models can improve the computational cost.

3. Structural recognition models are slow while statistical models have high speed due to the presence of high dimensional feature set. However, statistical models have high memory requirements.

4. An efficient combination strategy is required to combine the statistical and structural recognition models.

5. Deep learning based models suffers from the issue of vanishing gradient problem during learning of weights. Many researchers have solved the problem, but still much improvement is needed.

6. The optimization of hyper parameters is a major obstacle in the use of deep architectures. A vast room exist in research to find the optimization approaches for deep networks.

7. The high computational cost required for training of deep architectures should be improved with the help of some parallel algorithms.

### 2.2.7 Post-processing

Post processing has great significance in text recognition process as it ranks the results of classifier based on some confidence measures. This sections identifies the research issue for post processing of text recognized during recognition process.

1. The various confidence measures available performs almost in a similar manner. Hence, the selection of a particular confidence measure is a crucial task.

2. The lexicon size highly affects the speed and accuracy of recognition process. Large size will demand high memory and time for searching, while small size misses the recognition for common patterns.

3. Dynamic dictionaries are needed to handle words which do not belong to vocabulary. RNN models have performed well in recognizing such data.

4. Language models along with expert classifiers are needed to process confusing symbols.

# Chapter 3

# Proposed System: Bilingual HTR for Academic Domain (BHTRforAD)

This chapter presents the system of proposed bilingual HTR for Academic domain using text from Gurumukhi and Latin scripts. The architecture of BHTRforAD has been discussed in the following sections:

## 3.1 Architecture of BHTRforAD Framework

The researchers in the past used only the traditional machine learning approaches in the area of bilingual text recognition for Gurumukhi-Latin scripts. The proposed architecture is the first of its kind which recognizes Punjabi-English text along with numeric and special characters for academic domain data. This proposed system includes many sub processes of text recognition like segmentation, feature extraction, script identification, character recognition and generating the doc file of recognized text. It has combined deep learning and machine learning based approaches for feature extraction and classification processes.

### 3.1.1 Layered View

The architecture of BHTRforAD has been divided into four layers. Table 3.1 summarizes the modules and sub processes accumulated at each layer.

**Table 3.1:** Layers of BHTRforAD System.

| Layers | Modules | Sub Modules | Processes |
|---|---|---|---|
| Layer 1 | Data collection | Data pre-processing, Class labeling and balancing | binarization, normalization, labeling, resizing |
| Layer 2 | Segmentation | Line segmentation, Word segmentation, Character Segmentation | Touching line segmentation, intra word gap segmentation, composite character segmentation |
| Layer 3 | Processing of segmented data | Feature extraction, classification | Script identification, Gurumukhi composite character recognition, English character recognition, Alphanumeric and special character recognition |
| Layer 4 | Reporting Results | Constructing words from recognized characters | Generating the doc file corresponding to recognized data |

All the four layers, as mentioned in the Table 3.1 given above, have been described as follows:

1. **Layer 1:** This layer is responsible for data collection and pre-processing of data. Academic domain data has been collected in the form of handwritten documents. Further, the segmented text from documents is pre-processed to feed into recognition modules. Pre-processing module includes the binarization and labeling of data corresponding to classes.

2. **Layer 2:** It includes the segmentation of handwritten documents into sub parts like lines, words and characters. Documents may contain curved, skewed, and touching lines etc. Line segmentation module segments such lines; and further word segmentation is performed to get the individual words. The classifier takes the individual

characters for classification. Hence, character segmentation is performed to get the individual characters from words.

3. **Layer 3:** Handwritten documents contain bilingual text. To process such data, this module consists of script identification and the corresponding OCR for character recognition. Three character OCRs have been designed corresponding to scripts i.e. Gurumukhi OCR, English OCR and Alphanumeric OCR.

4. **Layer 4:** This layer generates the output of recognition model to doc file. It constructs the words from characters generated by recognition module.

### 3.1.2 Detailed Architecture

The novel architecture of Bilingual handwritten text recognition for Academic domain is presented in Fig 3.1. This section elaborates every module and sub process of the architecture.

#### 3.1.2.1 Data collection and Pre-processing

There is no benchmark dataset available for bilingual text recognition of Punjabi-English language in the academic domain. Thus, this research work is a modest attempt to design bilingual datasets for bilingual text recognition. Apart from it, some other available datasets have also been used in this work. All the datasets have been described as hereunder:

1. **Bilingual Handwritten Documents Dataset:** This dataset has been designed for academic domain considering the writers and content of documents from academic domain. Text in documents include alphanumerics, special characters, and data from Punjabi and English languages. It has 130 documents of varying character size, curved, skewed and touching lines. Further, many styles of document like multiple paragraphs, plain document, application type document, etc, have been considered.

2. **Composite Character Dataset for Punjabi:** For Gurumukhi script, a new dataset of composite characters has been proposed. Composite character is a combination of

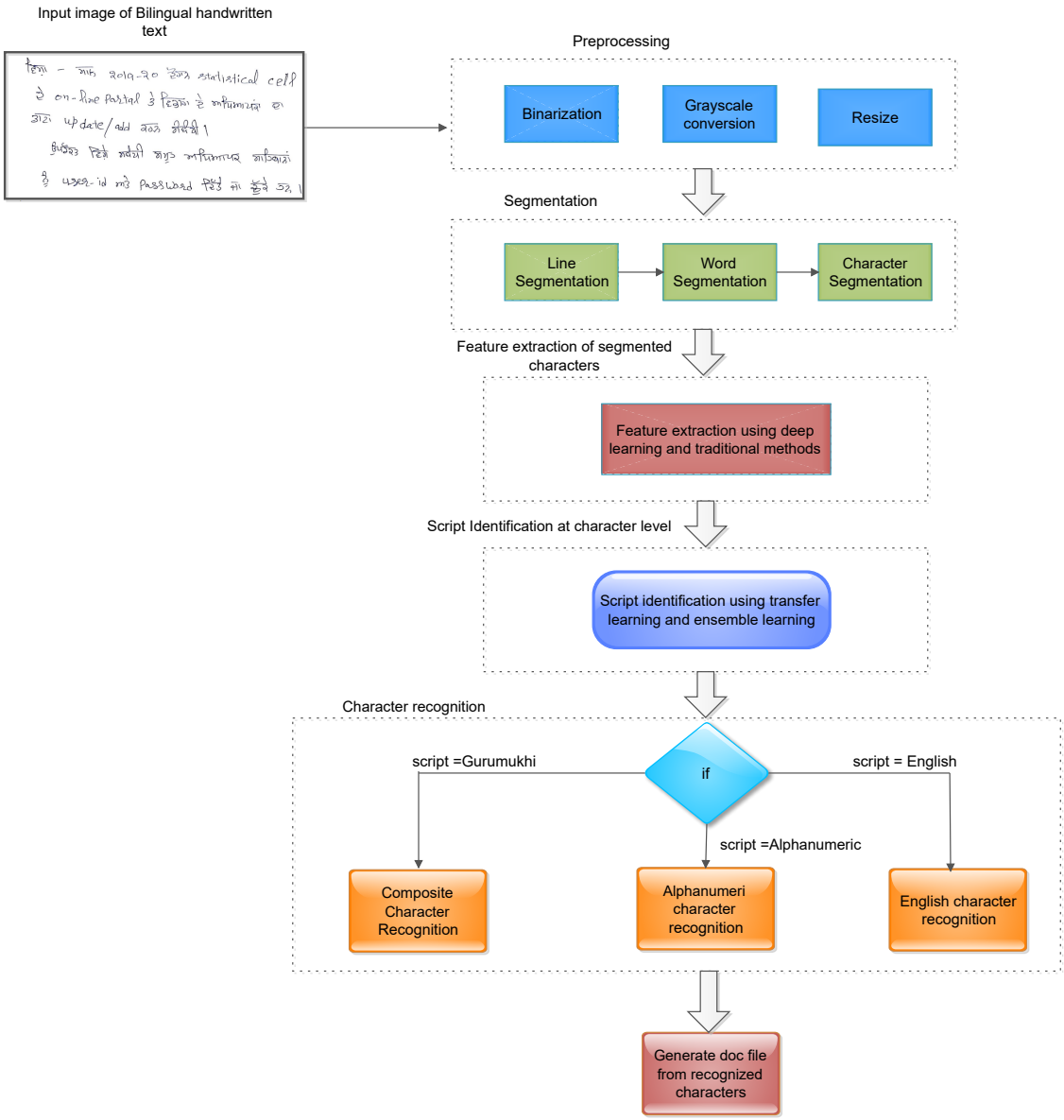



**Figure 3.1:** Architecture of proposed BHTRforAD system

character and vowel. It consists of 307 classes of composite characters resulted from the combination of 12 vowels and 35 characters.

3. **IAM Dataset:** To evaluate the proposed architecture, a benchmark dataset of handwritten documents called IAM has been considered. It consists of 1539 pages of handwritten text. It was designed in the year 1999. However, only 146 text pages have been considered for the purpose of this work.

4. **Pre-segmented handwritten character dataset:** It includes the presegmented handwritten characters from Gurumukhi, English characters and certain special characters.

The pre-processing of all the above mentioned datasets includes: gray scale conversion, binarization, resizing and class labeling. For pre-processing of data in the proposed system, we have used all the traditional methods. Otsu's method has been used for binarization of gray scale images. All the images in datasets have been resized corresponding to the used model requirements. Further labeling of each image in the dataset has been done.

#### 3.1.2.2 Segmentation

A heuristic based approach has been designed using projection profiles, structural and statistical properties of script to segment the handwritten documents at the line, word and character level. Projection profiles are the most commonly used approaches for segmentation of textual data. The results of segmentation have been refined further using some post-processing after each step. The proposed approaches are able to segment the bilingual text with straight, curved, skewed, and touching lines without any kind of pre-processing steps. The details about the line, word, and character segmentation are as follows:

1. **Line Segmentation:** Line segmentation approach has been designed using the projection profiles and statistical properties of text written. The binarized image is the input for line segmentation process. The following are the steps of line segmentation process:

   *Step 1:* Initially, the handwritten document is divided into horizontal stripes using projection profiles of the document. These stripes may include individual text lines, overlapped, touching text lines, modifiers present in Gurumukhi text and some part of text. Post processing of all resulted stripes will be performed to get the refined results.

   *Step 2:* To get the refined stripes, this step merges stripes with size less than threshold (includes modifiers or part of character) with the candidate stripe including text lines. An algorithm called merging_of_stripes has been designed to perform the merge operation. Stripes having more than 20 rows are the candidate stripe, i.e., one which includes the text lines. Stripes having less than 20 rows, contain lower and upper modifiers or some part of text lines. Such stripes need to be merged in candidate stripes.

Thus, the algorithm merging_of_stripes, merges the lower zone and upper zone stripes with middle zone.

*Step 3:* From the set of refined stripes, this step identifies the stripes having skewed, touching, and overlapped lines. It considers the average size of stripes to find the candidate lines. The stripes having a size more than the average size include the touching and overlapped text lines. Such stripes are the input for further processing, while the others are for word segmentation.

*Step 4:* In this step, the stripes found from Step 3 are divided into vertical slices. The size of each vertical slice is half of the number of columns in the stripe. To get the individual lines, projection profile is further calculated for each vertical slice.

Steps 1 to 3 are repeated for each stripe until individual lines are found.

*Step 5:* This step regenerates the whole text line. IT merges the vertical stripes vertically to get the required text line.

Figure 3.2 represents flow chart of line segmentation process.

2. ***Word Segmentation:*** Word segmentation is performed to get the individual words from lines segmented using line segmentation process. The following are the steps for word segmentation:

   *Step 1:* The proposed approach uses the inter word and intra word distance to segment the words of Gurumukhi and English scripts. Initially, vertical projections are calculated for a text line corresponding to the columns. Vertical Projections (VPs) have zero value for columns having no text, while non-zero value for columns containing the text.

   *Step 2:* For segmentation, the proposed approach looks for contiguous columns containing non zero VPs values. The boundaries of word are defined by the first column containing non zero value to first column containing zero value of VP after contiguous non-zero values of VP.

   *Step 3:* To segment the words with intra word gap, the average word size of all the words in the text line has been calculated. The word size provides the inter and intra word gap.

   *Step 4:* Dilation is performed using the inter and intra word gap found from Step 3.

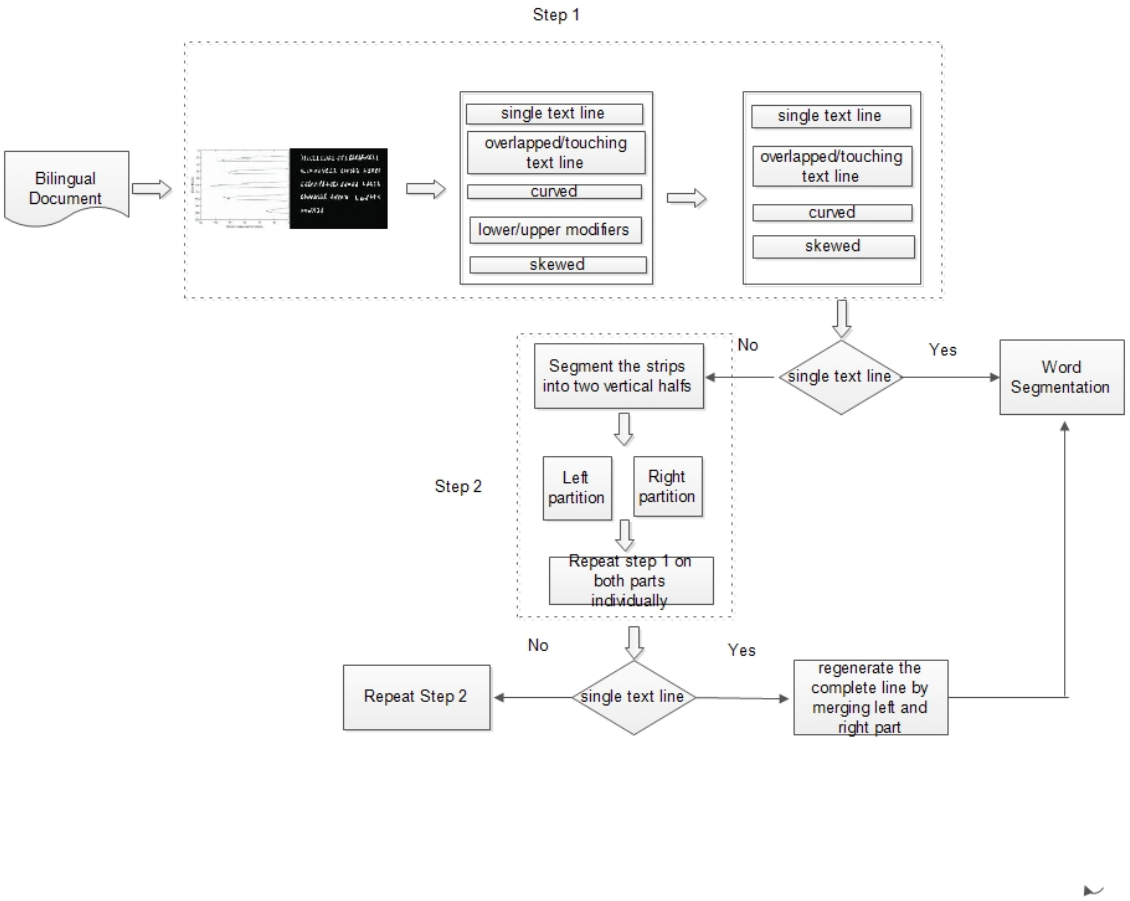



**Figure 3.2:** Flow chart of line segmentation process

Further, vertical projection is performed to get the boundaries of words.

*Step 5:* VP is performed to refine the errors of under segmented words. It helps in segmenting the closed and connected words.

Figure 3.3 represents the process of word segmentation.

3. ***Character Segmentation:*** Character segmentation is a challenging task for Gurumukhi and English scripts due to cursive writing. Characters in Gurumukhi are connected by header line, and in English are connected by lower baseline. It is difficult to find the header line and baseline in handwritten text as compared to the printed one. However, this work explores the recognition based character segmentation approach to segment the skewed and touching characters. This process results in composite characters for Gurumukhi script. The following steps have been used for character

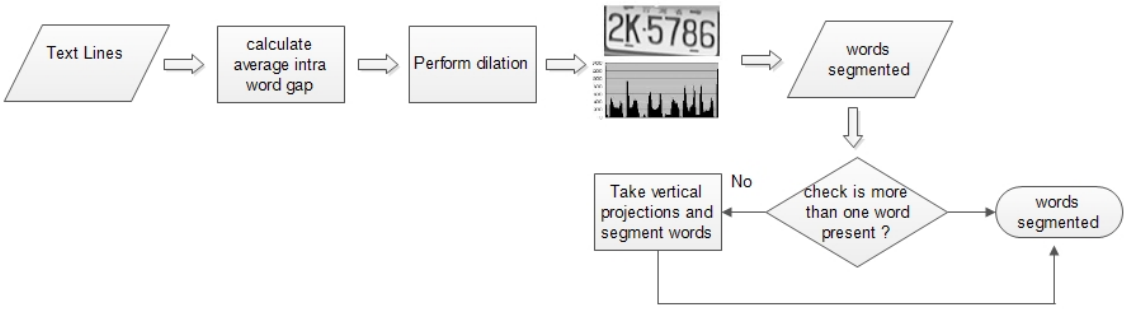



**Figure 3.3:** Flow chart of Word Segmentation Process

segmentation:

*Step 1:* Some writing styles have disconnected characters in a word. To segment such words, simple vertical projections are used. It results in characters and sub words.

**Step 2:** Further, baseline or header lines are identified to segment sub words. Rows with maximum horizontal projections have been considered to identify the header line or baseline. The vertical projections are taken from header lines or base lines to find the character boundaries.

*Step 3:* The results of Step 2 may include some over segmentations of characters. To get the actual character, vertical merging of segmented parts is performed. Further, recognition of resulted character is performed to validate the resultant character.

#### 3.1.2.3 Script Identification

Suppose, we have a set $H = \{h_1, h_2, ..., h_n\}$ of handwritten character images where each image $h_i = \{C, S\}$ has a character class belonging to a particular script. S is the set of scripts, while C is the set of characters belonging to that script. Thus, set S is defined as:

$S = \{S_1, S_2, S_3\}$

where, $S_1$= Gurumukhi script

$S_2$= English script

$S_3$= Numeric and special characters

Script identification defines classification function f on set of character images which assigns

the most accurate script from the set S to the character. Function f is defined as:

$$f : (h_i \rightarrow s_i | s_i \in S) \tag{3.1}$$

Figure 3.4 depicts the architecture of proposed script identification process.

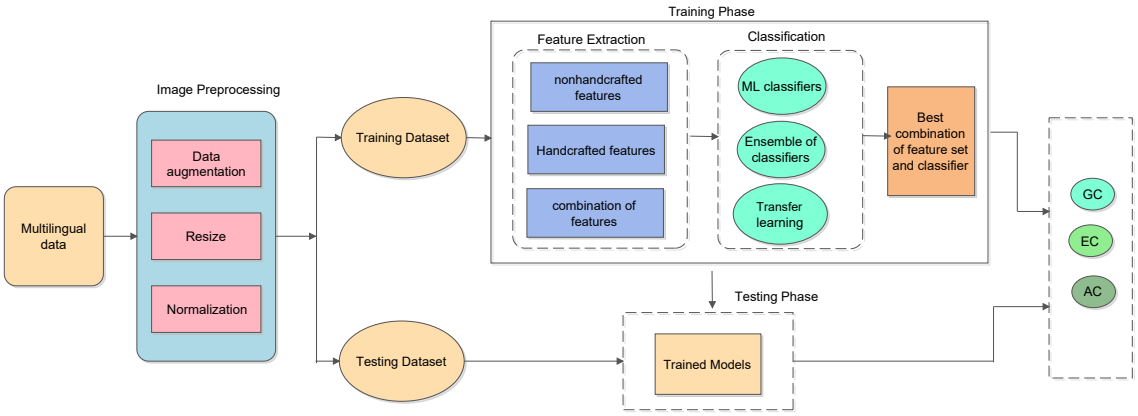



**Figure 3.4:** Architecture of Script Identification process

1. **Data Augmentation:** Data augmentation has been used to meet the demands of large data by deep neural networks. Two different techniques of data augmentation viz. rotation and flipping have been used. It helps in solving the class imbalance problem and making the model transformation invariant. Rotation operation has been performed using rotation of 20deg angle, while range of 0.2 has been used for zooming of images.

2. **Feature extraction:** Two types of features, viz. traditional features and deep learning based features have been extracted for script identification. Traditional feature extraction includes GLCM, Gabor and HOG features, while deep learning based features include LeNet5, Vgg19, and ResNet50 features. For deep learning features, the last convolution layer of trained network is used as a feature extractor. Further, many combinations of features have been designed to find the best combination. Out of three deep networks, LeNet5 features have been used for designing a combination of features. In all 15 combinations of features have resulted from traditional and deep learning features. Figure 3.5 shows the architecture of deep learning and traditional feature extraction methods.

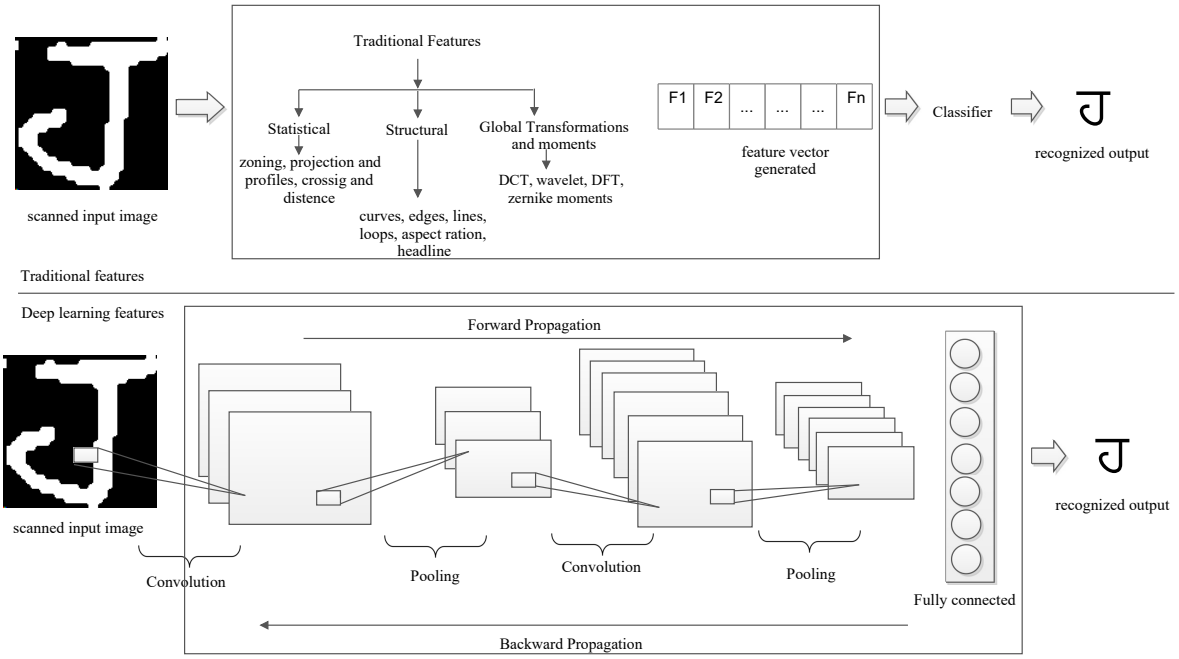



**Figure 3.5:** Architecture of handcrafted and non handcrafted features

3. **Classification:** Features extracted through traditional and deep learning methods, are used for classification using transfer learning, ensemble classifiers, and traditional machine learning classifiers. Three base classifiers, i.e., SVM, KNN, and decision tree are used for classification. From deep networks, pre-trained VGG19 and ResNet50 are used for classification. Further,an ensemble of classifiers has been designed using bagging, voting and boosting to improve the performance of classifiers .

#### 3.1.2.4 Character Recognition

After script identification, the next step is focused on choosing the required CR system from the pool of CR systems. For bilingual HTR, three CRs corresponding to scripts, i.e., Gurumukhi CR, English CR and special character and numeral CR have been designed as follows:

1. **Gurumukhi CR:** For Gurumukhi character a composite character recognition approach has been proposed to recognize the character and vowel together. For this, the composite dataset designed in the data collection phase has been used.The mathematical formulation of the proposed approach is as follows:
   Let P = $\{X'_{C\times M}, Y'_{C\times M}\}$ be set of labeled training data; where, $X'_{C\times M}$ is the set of

images of composite characters, i.e. combination of characters and modifiers, while $Y'_{C\times M}$ is the set of corresponding class of composite character (class of character and modifier). Here, $C = \{c_1, c_2, ..., c_{35}\}$ is the set of 35 characters of Gurumukhi; and $M = \{m_1, m_2, ...m_{12}\}$ is the set of 12 commonly used modifiers in Gurumukhi. $c \times m$ denotes all the combination of characters and modifiers which results into 307 combinations. The mathematical formulation of dataset is represented as :

$$X' = \{I_1, I_2, I_3..., I_n\} \tag{3.2}$$

$$Y' = \{C_{c_1,m_1}, C_{c_1,m_2}, ..., C_{c_2,m_1}, C_{c_2,m_2}, ...Cc_{35}, m_{12}\} \tag{3.3}$$

To recognize such data i.e. P two classifiers $R_1$ and $R_2$ have been designed to classify the images in X' into corresponding class from the set of class labels, i.e., Y'.

The classifier $R_1$ extracts features from dataset P corresponding to characters and classifies the data into character class. The function $f_1$ is a member of classifier $R_1$ which predicts the character class $\delta_d$ from set C corresponding to inupt image $x$ from set $X'$. The mathematical representation of $f_1$ is as:

$$f_1 : (x- > \delta_d \mid \delta_d \in C) \tag{3.4}$$

Similarly classifier $R_2$ extracts features of dataset corresponding to modifiers and classifies into modifier classes. The function $f_2$ is a member function of classifier $R_2$. It maps the input image $x$ from set $X'$ into corresponding modifier class i.e. $\sigma_d$ from set M. The mathematical representation of $f_2$ is as :

$$f_2 : (x- > \sigma_d \mid \sigma_d \in M) \tag{3.5}$$

The final output is the combination of $R_1$ and $R_2$ i.e. character and modifier. For an image from dataset $X'$ the classifier 1 with function $f_1$ will provide the character class and the classifier 2 with function $f_2$ will generate the modifier class for the image. Thus, the final output label for composite character image will be generated by merging the

output of both classifiers. Figure 3.6 shows the architecture of composite character recognition.

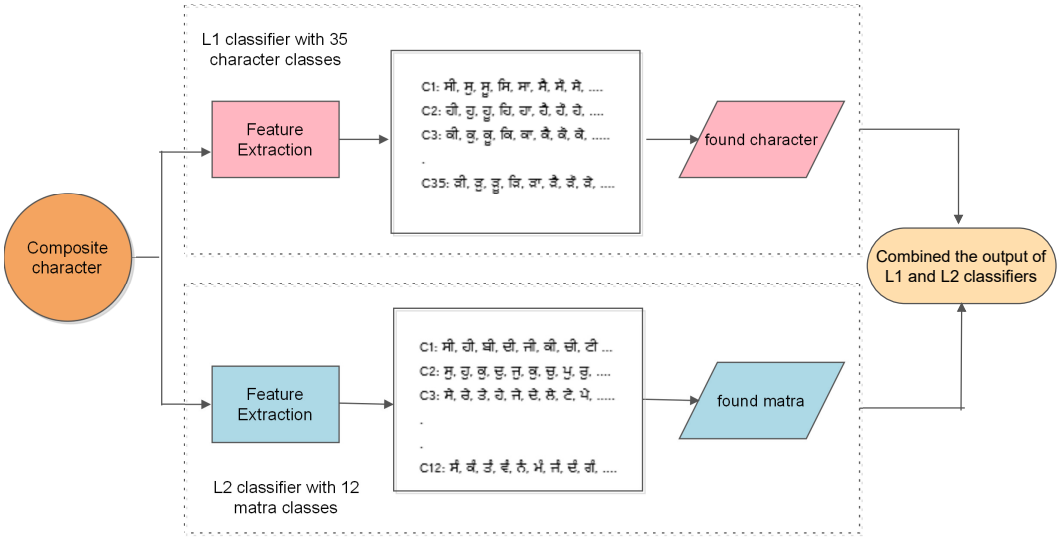



**Figure 3.6:** Composite Character Recognition

2. **English CR:** In this architecture, another dataset Q=$\{X'_c, Y'_c\}$ of presegmented characters English script has been considered. Here, set $X'_c$ is the set of images of presegmented characters; and $Y'_c$ is the set of class labels corresponding to the characters. The final dataset Q consist of 52 classes having presegmented characters and vowels set of Gurumukhi script such as:

   X'=$\{C_1, C_2, ..., C_n\}$

   Y'=$\{L_1, L_2, ..., L_{52}\}$

   Now, for an image of presegmented character from $X'$ dataset, a single function $f_3$ has been designed to classify it into a corresponding class label belonging to set $Y'$, i.e.,

$$f_3 : (q- > \rho'_X \mid \rho'_X \in Y') \tag{3.6}$$

3. **Alphanumeric CR:** In this architecture, another dataset R=$\{X'_c, Y'_c\}$ of presegmented numerals and special characters has been considered. Here, set $X'_c$ is the set of images of presegmented numerals and special characters; and $Y'_c$ is the set of class labels corresponding to numerals and special characters. The final dataset R consist of 22 classes having presegmented characters and vowels set of Gurumukhi script such as:

X'=$\{C_1, C_2, ..., C_n\}$

Y'=$\{L_1, L_2, ..., L_{22}\}$

Now, for an image of presegmented character from $X'$ dataset, a single function $f_4$ have designed to classify it into a corresponding class label belonging to set $Y'$, i.e.,

$$f_4 : (r- > \rho'_X \mid \rho'_X \in Y') \tag{3.7}$$

#### 3.1.2.5 Generating Doc File

The recognized results of OCRs are stored in the word document. The words are constructed from the recognized characters. Further, a doc file has been generated to store the recognized text in the files. This is an editable file which makes the handwritten text editable.

# Chapter 4

# Design and Implementation

This chapter includes the details of design and implementation of BHTRforAD system. The various Unified Modeling Language diagrams used in designing of framework are presented well in detail. Further, the experimental and algorithms details are explained in this chapter.

## 4.1 Design

UML (Unified Modeling Language) is designed by Object Management Group in 1997 to visualize the complex behaviors of software. It helps in documenting and constructing the various aspects of framework. UML is classified into tow major categories i.e. structural and behavior modeling.

### 4.1.1 Structural Modeling

Structural modeling represent the static view of the system using Class diagram and component diagrams. These diagrams depict the different elements of the proposed system and idea of assembling these elements.

#### 4.1.1.1 Class Diagram

Figure 4.1 depicts the class diagram of BHTRforAD system. The static view of the framework is represented using various classes along with attributes and functions corresponding to each class. Pre-processing class performs binarization and grayscale conversion of the

uploaded document image. At a time only one document image can be uploaded. The segmentation class applies line, word and character segmentation function image to get the individual characters. Further the script identification and OCR classes uses feature and classifier classes to identify the script used and character from that script. Keras and SCIpy packages are used by feature and classifier classes to perform various functions like traditionalfeature(), DeeplearningFeature() etc. The recognized characters are used by GeneratingOutput class to construct a doc file using function like Constructingwords() and savingdocfile().

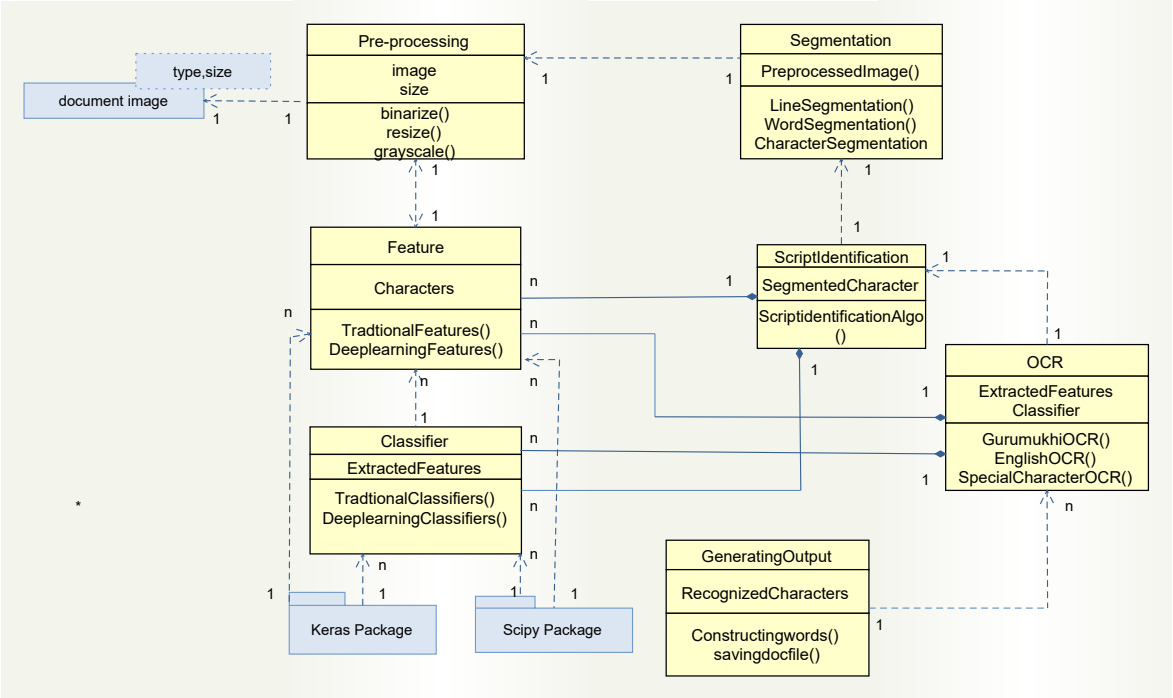



**Figure 4.1:** UML class diagram

#### 4.1.1.2 Component Diagram

The second most popular diagram used to represent the static view of the system is component diagram. It includes the various libraries, packages and executable scripts. Figure 4.2 depicts the component diagram of BHTRforAD system. Varous matlab and python scripts used by different components to implement the functionality of system are presented.

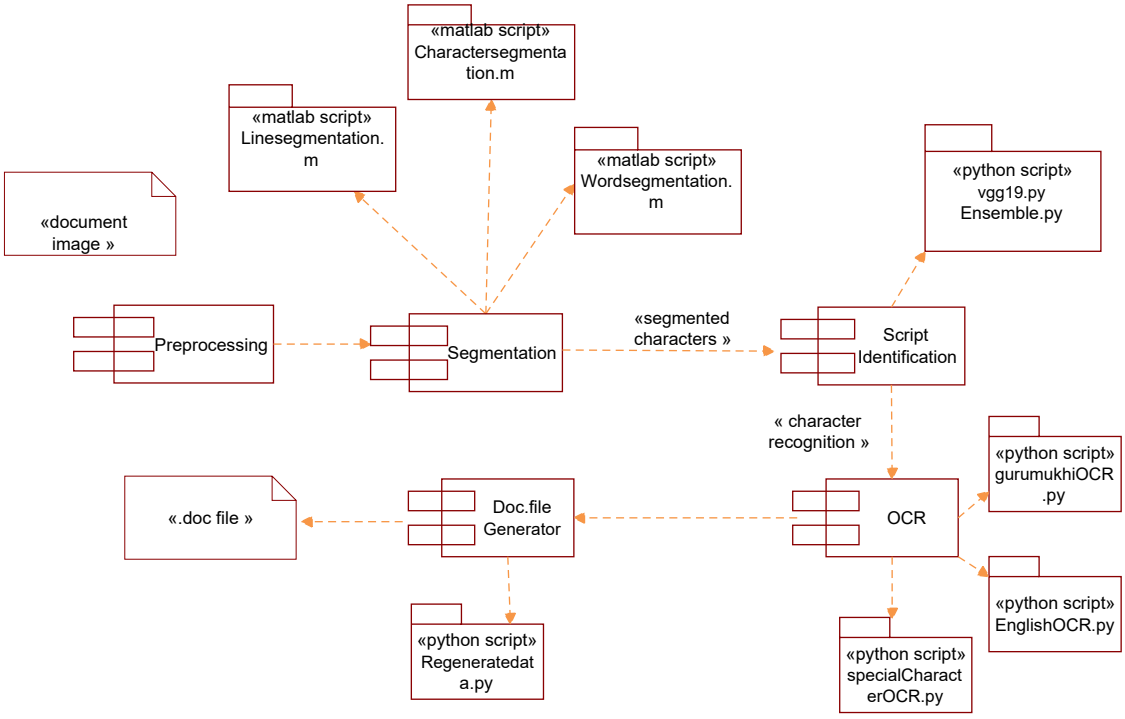



**Figure 4.2:** UML component diagram

### 4.1.2 Behavioral Modeling

To implement the dynamic behavior of the system, behavior modeling is considered in UML. It includes various diagrams like statechart, usecase and sequence diagram etc. which are explained in the following section:

#### 4.1.2.1 Usecase Diagram

Use case diagrams represent the dynamic behavior of system using some internal and external factors related to system. The three main components of use case diagram are: usecases, actors and their relationship. Actor is the user of the system while usecases are the various processes in the system which interact with each other to complete the functionality of system. Figure 4.3 represents the boundary of framework where various usecases are presented while anything present outside this boundary is considered as actors. In BHTRforAD system, an actor uploads image, request for pre-processing, segmentation, script identification, OCR and doc file creation usecases. Usecase diagrams are helpful in identifying the

functional requirements of the system and relationship between various internal and external factors.

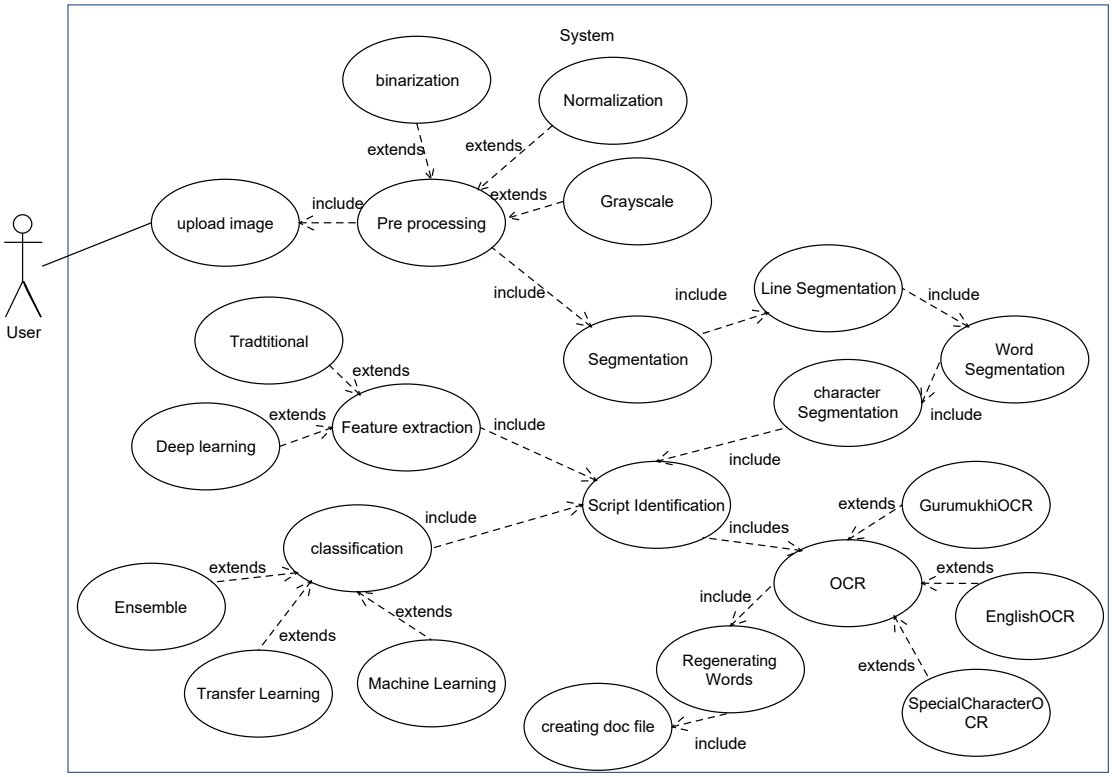



**Figure 4.3:** UML usecase diagram

#### 4.1.2.2 Sequence Diagram

The focus of sequence diagrams is to depict the interactions occurring in the proposed framework. Figure 4.4 represents the lifelines of various objects along with their interactions occurring with each other. In the proposed framework, a pre-processing module initializes the process and pre-processed image is sent to segmentation module which sends back the segmented characters for pre-processing. Further resized character is sent to script identification module which identifies the script from trained models and sends the character to OCR for recognition. The recognized character from OCR is sent as input to word generation module. The final output is saved in doc file.

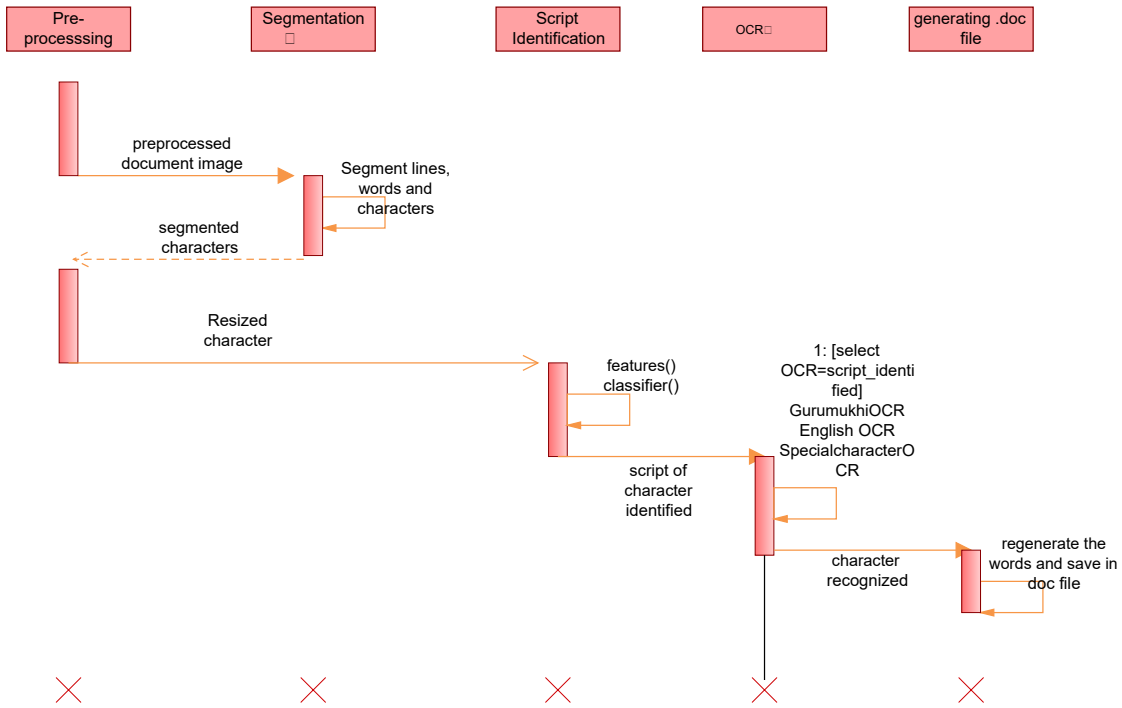



**Figure 4.4:** UML sequence diagram

#### 4.1.2.3 Statechart Diagram

Statechart diagrams are used to represent the different states adopted by components of the framework. Figure 4.5 depicts the statechart diagram of the proposed framework. The aim of this diagram is to represent the flow of control between various states of components. The major states in BHTRforAD system are to upload the image of document, binarize and resize the image. After that, various states during segmentation process, feature selection and classification.

## 4.2 Implementation

This section discusses the implementation details for bilingual handwritten recognition system under various sub-processes including experimental setup and datasets.

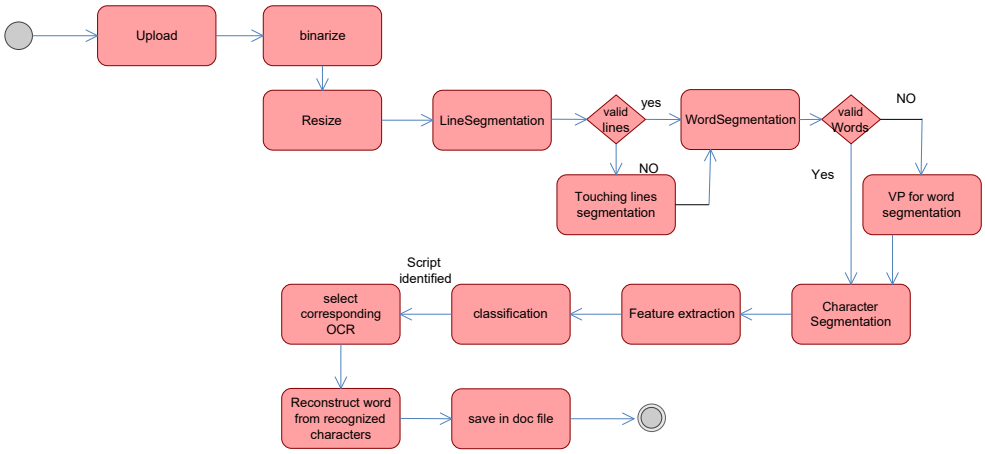



**Figure 4.5:** UML statechart diagram

### 4.2.1 Experimental Details

The proposed system has been implemented using two languages i.e. Python and Matlab. Segmentation processes are implemented in Matlab, while python hs been used for feature extraction and classification.

To implement the deep learning modules for feature extraction and classification, Keras framework has been used. It includes number of pre-trained deep learning models. Similarly, for simple machine learning algorithms, scipy library of python has been used. To design the framework for recognition system, matlab designer and jupyter notebook are considered.

### 4.2.2 Pre-processing and Segmentation Algorithms

These segmentation algorithms has been implemented in matlab language. Segmentation of handwritten documents is divided into three sections i.e. line, word and character segmenta-

tion. Each algorithm uses preprocessing techniques to clean the input data.

#### 4.2.2.1 Line Segmentation

Line segmentation process includes various sub-process to segment the touching, merged, closed, separated and curved text lines. Algorithm 4.1 segments lines initially using projection profiles. The in depth details of the various steps used by algorithms are as follows:

Step 1: The first step is to perform the projection profiles on document image. It gives us the horizontal stripes (as step 1 to 5 in algorithm 4.1).
Step 2: The next step is to find the stripes which contains text lines (step 6) and get the boundaries of lines (step 7).
Step 3: Now many stripes will contain partial data, te get the accurate results we perform a process called merging of horizontal stripes.
Merging of horizontal stripes:
It calculate the size of stripes and the one which has threshold value less then 20 will be concatenated with the immediate adjacent to that stripe but having threshold value greater then 20.
The next step is to segment the touching lines or stripes containing more than one text line (Algorithm 4.3).
Step 4: To segment the touching and overlapping text lines we will divide the stripe to two vertical segmentation which contains half the number of columns to the original stripe (as in step 2 to 6).
Step 5: Now perform step 1 to 3 to each individual slice got in step 4.
Step 6: To get the original line we will regenerate the stripes using slices from step 5. In this we concatenate the individual slices from both the slices.

The algorithm 4.1 calls algorithm 4.2 to perform the post processing of segmented text lines. It refines the results. Now to segment the skewed, touching or curved text lines it calls algorithm 4.3.

**Algorithm 4.1** Line Segmentation of Bilingual handwritten documents

**Require:** HBT:Handwritten bilingual text image of size row × col
**Ensure:** segmented text lines

I ← imread(HBT)
gray ← gray_scale_coversion(I);
binary ← gray_to_binary_conversion(gray);
binary1 ← invert(binary) ▷ invert image to take horizontal projections
$proj_{binary1} = \sum_{i=1}^{rows} \sum_{j=1}^{cols} \text{if}(binary1(i,j) == 1)$; ▷ calcaulte the horizontal projections
lines ← $proj_{binary1} \geq 1$ ▷ Find rows containing text data
d← diff(lines) ▷ find the rising and falling edges of rows with text
startingcol ← find(d≥0)
endingcol ← find(d≤0)
stripes ← lines; ▷ extract each line and save it
refined_stripes ← merging_horizontal_stripes(stripes); ▷ call Algorithm 4.2 and find the stripes with touching, skewed or curved text
**for** refined_stripes **do**
    $avg_{stripsize} \leftarrow average(size(refined\_stripes))$ ▷ the stripes with size greater than average size will contain touching and overlapping lines
    **for** refined_stripes **do**
        **if** size(refined_stripes[i])≥ $avg_{stripsize}$ **then**
            stripes_for_vertical_seg ← refined_stripes
            Go to touching_line_segmentation(refined_stripes) ▷ Call algo 4.3
        **end if**
    **end for**
**end for**

**Algorithm 4.2** merging_of_stripes

**Require:** stripes
**Ensure:** refined_stripes

**for** stripes **do**
    strip1 ← stripes[i]
    strip2 ← stripes[i+1]
    **if** r1≥ 20 and r2≤ 20 **then**
        refined_stripes ← concatenate_horizontally(strip1, strip2)
    **else if** r1≤ 20 and r2≥ 20 **then**
        refined_stripes ← concatenate_horizontally(strip2, strip1)
    **else**
        refined_stripes ← stripe1
    **end if**
**end for**

**Algorithm 4.3** touching_line_segmentation

**Require:** stripes_for_vertical_segmentation
**Ensure:** segmented text lines

```
touching_lines ← stripes_for_vertical_segmentation
for touching_lines do
    rows1,cols1 ← size(touching_lines)
    slice1 ← touching_lines(:,1:cols1÷ 2);
    slice2 ← touching_lines(:,cols1÷ 2 + 1: cols1);
    repeatStep 5 to 17 of Algorithm 1
    until we finds stripes with single text line ▷ save segmented text lines of slice 1 and slice 2 in two different folders for further processing
    lines_slice1 ← segmented text lines of slice 1
    lines_slice2 ← segmented text lines of slice 2  ▷ To form the complete text lines we need to merge the corresponding segmented lines from both slices
    for lines_slice1 do
        line1_slice1 ← line_slice1[i]
        line1_slice2← line_slice2[i]
        r1,c1 ← size(line1_slice1)
        r2,c2 ← size(line1_slice2)
        if r1==r2 then
            line ← concatenate_horizontally(line1_slice1,line1_slice2)
        else
            resize images and concatenate
        end if
    end for
end for
```

#### 4.2.2.2 Word Segmentation

Word segmentation has been implemented using algorithm 4.4. The step by step breakdown of the algorithm is as follows:

Step 1: The input to this algorithm is the output of the line segmentation algorithm. To get the word boundaries for dilation this algorithm calculates the intra word gap. For this it uses average word size of the words in line and using that size we will perform dilation (step 3-4).

Step 2: The next step is to get the boundaries of the word. We have used vertical projection profiles to get the boundaries (as in step 6-7).

Step 3: The columns having contiguous non zero values for projection profiles are the one which contains text (as in 7-10).

**Algorithm 4.4** end-point detection algorithm

**Require:** HTL: handwritten text lines containing bilingual words
**Ensure:** segmented words

1: I ← imread(HTL) ;
2: binary ← gray_to_binary_conversion(gray);
3: avg_word_gap ← calculate_intra_word_gap(binary);
4: img_dilate ← imdilate(binary); ▷ dilate image corresponding to word gap found
5: binary1 ← invert(img_dilate); ▷ invert image to take vertical projections
6: $Vproj_{binary1}$=$\sum_{j=1}^{cols}\sum_{i=1}^{rows}$if(binary1(j,i)==1); **comment:** calcaulte the vertical projections
7: words ← $proj_{binary1} \geq 1$ ▷ Find cols containing text data
8: d← diff(words) ▷ find the rising and falling edges of rows with text
9: startingcol ← find(d≥0);
10: endingcol ← find(d≤0);
11: words ← words; ▷ extract each word and save it

#### 4.2.2.3 Character Segmentation

Character segmentation of words has been performed using following algorithm 4.5. The step by step breakdown of the algorithm is as follows:

Step 1: The first step is to get the vertical projections of the words. As some words here include subwords and disconnected characters (as in step 1 to 4 in algorithm).

Step 2: The next step is to find the header lines of baselines in the text where we use horizontal projection profiles. The rows with maximum horizontal projections represents the header line or baseline (as in step 5).

Step 3: Further vertical projections are taken from header line or baseline to get the character boundaries (as in step 6-7).

Step 4: The results of previous step will include some oversegmentations. Hence, we use recognition based technique where we will recognize each vertical segmentation, manually to get the actual character. We merges the vertical segmentation until we cant find the exact character. This is the manual process which can be automated in future using some recognition models (as in step 8-11).

**Algorithm 4.5** character segmentation algorithm based on segmentation recognition process

**Require:** HW: segmented words
**Ensure:** segmented characters
1: I ← imread(HW) ;
2: binary ← gray_to_binary_conversion(gray);
3: VP ← calculate_vertical_projections(binary); ▷ Find the disconnected characters and subwords
4: character_and_subwords ← word_segmentation(VP); ▷ segment the word using VP
5: baseline ← find base line(character_and_subwords); ▷ To segment the sub words
6: The colomns with lesser number of black pixels are the candidate points for segmentation
7: segmented_characters(n) ← baseline based segmentation ▷ It will give over segmentations
8: To find the exact character perform merging of segmented results until we finds the recognized character
9: **for** i $\leq$ n **do**
10: char ← merge(char(i), char(i+1),...n) ▷ manually check for the character and stop merging
11: **end for**

### 4.2.3 Script Identification

Script identification process includes feature extraction and classification tasks. Three traditional and three deep learning based features have been extracted. Further various combinations of extracted features are designed. The pseudocode for features and classifier used in the work is explained as follows:

*>Load dataset*

*>get the training labels*

*>create two array i.e. feature_array[] for features and label[] for labels*

*>Read images from training folder along with labels*

Extract Gabor features

*>F1= Gabor features*

Extract GLCM features

*>F2= GLCM features*

Extract HOG features

*>F3= HOG features*

Extract LeNet features

To extract the features use the trained model. The last convolutional layer of trained model has been used as feature extractor.

*>F4= LeNet features*

Design various combinations of features

Combine HOG and GLCM features GLCM features

*>F5= Concat(F2, F3)*

Combine Gabor and HoG features features

*>F6= Concat(F1, F3)*

Combine Gabor and GLCM

*>F7= Concat(F1, F2)*

Combine Gabor,GLCM, HOG features

*>F8= Concat(F1, F2, F3)*

Combine Gabor, GLCM and LeNet

*> F9=(F1, F2, F4)*

Combine Gabor, GLCM, LeNet, HOG

*>F10= (F1, F2, F3, F4)*

Combine Gabor, HOG, LeNet

*>F11= (F1, F3, F4)*

Combine GLCM, LeNet, HOG

*>F12= (F2, F3, F4)*

Combine LeNet and Gabor

*>F13= (F1, F4)*

Combine LeNet and GLCM

*>F14= (F2, F4)*

Combine LeNet and HOG

*>F15= (F3, F4)*

For classification: Three types of experiments have been implemented i.e. using three machine learning classifiers. using ensemble based classifier and using transfer learning for classification. The pseudocode for all three approaches is explained as:

1. Three machine learning based classifiers i.e. SVM , RF and KNN has been used to train each feature set. Hence it results into 45 classifiers.
   for F(i) in (F1 to F15) do:
   (a) train SVM classifier
   (b) train RF classifier
   (c) train KNN classifier
   end

2. Further for all feature sets ensemble classifiers like Voting, bagging and boosting have been implemented.
   for F(i) in (F1 to F15) do:
   (a) train Voting based classifier with base classifier (SVM, RF and KNN)
   (b) train Bagging based classifier with base classifier (SVM, KNN, RF)
   (c) train Boosting based classifier with base classifier (SVM, KNN, decision tree)
   end

3. Transfer learning based classifiers have been designed using Vgg19 and ResNet50. The pseducode for ResNet50 classifier is as :

   *>import Keras*

   *>import ResNet50*

   *>import ImageDataGenerator*

   Preprocess data corresponding to ResNet50

   *resize dataset images*

   *>Load training dataset or features with labels*

   Divide the dataset into training and testing

   *>traindata, testdata←train_test_split(feature,label)*

   perform one hot encoding of labels

   *>labels←(to_categorical(labels,no_of_classes))*

   *>Perform Data augmentation using rotation, flipping, and zooming operations*

   Initialize ResNet50 model trained on imagenet dataset excluding the top layer

   *>call ResNet50(include_top = False, weights = 'imagenet', input_shape = (224,224,3), classes = len(labels))*

   *>Fine tune the ResNet50 model corresponding to our dataset by adding dense and dropout layers*

   Add final dense layer as classification layer

   *>add dense(5,softmax)*

   Define the hyperparameters like epochs, learning rate, batch size

   Compile the model

   *>model.compile(optimizer, loss, metrics)*

   Train the model

   *>model.fit(generated_data, epochs, batch size)*

   Evaluate the model on test data

   *>model_model.predict(testdata)*

   The pseducode for Vgg19 classifier is as :

   *>import Keras*

*>import VGG19*

*>import ImageDataGenerator*

Preprocess data corresponding to VGG19

*>resize dataset images*

*>Load training dataset or features with labels*

Divide the dataset into training and testing

*>traindata, testdata←train_test_split(feature,label)*

Perform one hot encoding of labels

*>labels←(to_categorical(labels, no_of_classes))*

*>Perform Data augmentation using rotation, flipping, and zooming operations*

Initialize VGG19 model trained on imagenet dataset excluding the top layer

*>call VGG19(include_top = False, weights = imagenet, input_shape = (224,224,3), classes = len(labels))*

*>Fine tune the VGG19 model corresponding to our dataset by adding dense and dropout layers*

Add final dense layer as classification layer

*>add dense(5,softmax)*

Define the hyperparameters like epochs, learning rate, batch size

compile the model

*>model.compile(optimizer, loss, metrics)*

Train the model

*>model.fit(generated_data, epochs, batch size)*

Evaluate the model on test data

*>model_model.predict(testdata)*

### 4.2.4 OCR

After script identification the corresponding OCR will be selected. Hence, we have implemented three different OCRs w.r.t scripts. For character recognition we have implemented

VGG19 as classifier. The implementation details of the OCRs are given as:

#### 4.2.4.1 Gurumukhi OCR

Gurumukhi OCR is implemented using two VGG19 networks to recognize the composite characters from 302 classes. The pseducode of composite CR is as follows:

*>import Keras*

*>import VGG19*

*>import ImageDataGenerator*

Divide the dataset into two types: It will reduces the class size of dataset from 302 to 35 and 12. Type 1 dataset has 35 classes coresponding to character while Type 2 dataset has 12 classes corresponding to matras.

*>resize both dataset images*

*>Load Type 1 training dataset labels*

Divide the dataset into training and testing

*>traindata1, testdata1←train_test_split(Type1,label1)*

Perform one hot encoding of labels

*>labels1←(to_categorical(labels,no_of_classes))*

*>Perform Data augmentation using rotation, flipping, and zooming operations*

Initialize VGG19 model trained on imagenet dataset excluding the top layer

*>call VGG19(include_top = False, weights = imagenet, input_shape = (224,224,3), classes = 35)*

*>Fine tune the VGG19 model corresponding to our dataset by adding dense and dropout layers*

Add final dense layer as classification layer

*>add dense(35,softmax)*

Define the hyperparameters like epochs, learning rate, batch size

Compile the model

*>model1.compile(optimizer, loss, metrics)*

Train the model1

*>model1.fit(generated_data, epochs, batch size)*

Evaluate the model1 on test data

*>character_class←model.predict(testdata1)*

*>Load Type 2 training dataset labels*

Divide the dataset into training and testing

*>traindata2, testdata2←train_test_split(Type2,label2)*

Perform one hot encoding of labels

*>labels2←(to_categorical(labels2,12))*

*>Perform Data augmentation using rotation, flipping, and zooming operations*

Initialize VGG19 model trained on imagenet dataset excluding the top layer

*>call VGG19(include_top = False, weights = imagenet, input_shape = (224,224,3), classes = 12)*

*>Fine tune the VGG19 model corresponding to our dataset by adding dense and dropout layers*

Add final dense layer as classification layer

*>add dense(12, softmax)*

Define the hyperparametrs like epochs, learning rate, batch size

Compile the model

*>model2.complie(optimizer, loss, metrics)*

Train the model2

*>model2.fit(generated_data, epochs, batch size)*

Evaluate the model2 on test data

*>matra_class←model.predict(testdata2)*

Combine the output of these models

*recognised_charctaer←concatenate(charcater_class,matra_class)*

#### 4.2.4.2 English OCR

: The pseudocode to implement English OCR has been explained as below: *>import Keras*

*>import VGG19*

*>import ImageDataGenerator*

*>resize dataset images*

*>Load training dataset labels*

Divide the dataset into training and testing

*>traindata, testdata←train_test_split(data,labels)*

Perform one hot encoding of labels

*>labesl←(to_categorical(labels,no_of_classes))*

*>Perform Data augmentation using rotation, flipping, and zooming operations*

Initialize VGG19 model trained on imagenet dataset excluding the top layer

*>call VGG19(include_top = False, weights = imagenet, input_shape = (224,224,3), classes = 52)*

*Fine tune the VGG19 model corresponding to our dataset by adding dense and dropout layers*

*Add final dense layer as classification layer*

*>add dense(52,softmax)*

Define the hyperparametrs like epochs, learning rate, batch size

Compile the model

*>model.complie(optimizer, loss, metrics)*

Train the model

*>model.fit(generated_data, epochs, batch size)*

Evaluate the model on test data

*>character_class←model.predict(testdata)*

#### 4.2.4.3 Alphanumeric OCR

To recognize the alphanumeric data, alphanumeric OCR has been designed. The following pseducode implements the alphanumeric OCR: *>import Keras*

*>import VGG19*

*>import ImageDataGenerator*

*>resize dataset images*

*>Load training dataset labels*

Divide the dataset into training and testing

*>traindata, testdata←train_test_split(data,label1)*

Perform one hot encoding of labels

*>labels←(to_categorical(labels,no_of_classes))*

*>Perform Data augmentation using rotation, flipping, and zooming operations*

Initialize VGG19 model trained on imagenet dataset excluding the top layer

*>call VGG19(include_top = False, weights = imagenet, input_shape = (224,224,3), classes = 22)*

*>Fine tune the VGG19 model corresponding to our dataset by adding dense and dropout layers*

*>Add final dense layer as classification layer*

Define the hyperparameters like epochs, learning rate, batch size

compile the model

*>model.compile(optimizer, loss, metrics)*

Train the model

*>model.fit(generated_data, epochs, batch size)*

Evaluate the model on test data

*>character_class←model.predict(testdata1)*

### 4.2.5 Generating doc file

To generate the doc file, the pseducode is as following:

Store the recognized characters of a word in an array

*>character[]←character_class*

*>open a doc file*

*>Read an array character[]*

*>for i =1 to size(character[])*:

*>write in doc file*

*>end*

*>save file*

# Chapter 5

# Test and results analysis

This chapter presents the result and testing analysis for various phases of handwritten text recognition system. Further, the proposed BHTRforAD system has been tested using different kinds of handwritten documents. The proposed segmentation algorithms are tested for two different types of datasets corresponding to line, word and character segmentation. Further, the proposed algorithm for script identification has been evaluated. For Gurumukhi composite characters, the proposed approach has been tested on proposed composite character dataset. To test the robustness of script identification model and character recognition, K fold cross validation testing has been used. The various parameters like accuracy, precision, recall, f score have been considered. To compare the performance of proposed methods with state-of-the-art methods a comprehensive performance evaluation is performed.

## 5.1 Result analysis corresponding to various phases of BHTR

This section details the various results corresponding to each phase of handwritten text recognition. Various types of datasets like IAM, composite character, bilingual handwritten, and MNIST have been considered.

### 5.1.1 Performance Measures

This section discusses the various performance metrics used in the research work like visual criteria, detection rate, recognition accuracy and F1 measure etc. The importance of these metrics lies in their ability to evaluate the performance of algorithms. Detection rate we also call it recall, measures the proportion of true positives (correctly detected segments) out of all actual segments in the text. A high detection rate is crucial for ensuring that no important segments are missed during the segmentation process. As in document segmentation we have to take care of all these minor segmentation which can change the meaning of text written if missed out. Recognition accuracy measures the proportion of true positives (correctly detected segments) out of all detected segments by the algorithm. It indicates how precise the segmentation algorithm is in identifying relevant segments without including irrelevant ones. High recognition accuracy is important for ensuring the quality and relevance of the segmented output. It's particularly crucial in applications where precision is paramount, such as natural language understanding or sentiment analysis. The F1 measure is the harmonic mean of detection rate and recognition accuracy. It provides a single metric that balances both precision and recall, offering a comprehensive evaluation of the segmentation algorithm's performance. A high F1 score indicates that the algorithm achieves both high detection rate and high recognition accuracy simultaneously, striking a balance between completeness and precision in segmenting the text.

These metrics are relevant in our research to give the more comprehensive evaluations of the proposed algorithms. Different metrics capture different aspects of performance. Detection rate measures the algorithm's ability to find all relevant segments, recognition accuracy measures its precision in identifying relevant segments, and F1 measure combines both precision and recall.

These are very important to know the algorithm behavior as For example, if an algorithm achieves high recognition accuracy but low detection rate, it may be overly selective in segmenting the text, missing important segments. Understanding these nuances helps in refining and optimizing the algorithm for better performance. It helps for future analysis of the algorithms. Many other metrics are ROC, AUC, cross validation Segmentation level evaluations

etc which are not so relevant to our domain. As for segmentation we are not training any model so there is no need of cross validation and ROC metrics. Further segmentation level can be used in the future works to evaluate the segmentation of sentences and phrases.
For script identification tasks we have considered accuracy, time and feature vector metrics. As to differentiate between deep learning and traditional approaches these are the most relevant metrics. For composite character recognition again, I have used precision, recall, F1 measure and accuracy, computational time. As in these algorithms we are making comparison between deep learning approaches and traditional machine learning approaches. Hence, for this comparison, the most appropriate parameters are accuracy and computation time. Apart from these we have specificity, ROC, AUC, precision recall curve etc. Precision-recall curve and AUC-PR provide an alternative to ROC analysis, particularly when dealing with imbalanced datasets. Specificity is particularly useful when the cost of false positives is high, and correctly identifying true negatives is critical. Moreover in the literature highly used metrics are precision, recall and accuracy.

### 5.1.2 Segmentation Results

The evaluation of segmentation algorithms is mostly done using visual criteria by many researchers on the publicly available datasets. This work considers many evaluation strategies apart from visual criteria which are similar to the one used in many document segmentation competitions like ICDAR 2003, 2005, 2007, 2009 and some research articles [23, 35, 36, 38, 41]. These measures compare the area detected by algorithm and the area of ground truth. The match is called one to one is the matching score for this pair is equal to or above the evaluators acceptance threshold. From this match, we have considered three parameters named: detection rate, recognition accuracy and F1 measure (it is a global performance metric found by combining DR and RA ). These measures are calculated as follows:

$$DR = o2O/N \tag{5.1}$$

$$RA = o2o/M \tag{5.2}$$

$$F1measure = (2DRRA)/(DR+RA) \tag{5.3}$$

Here N is the number of text pixels in ground truth segments and M is the number of detected or resulted text pixels while o2o matching defines the number of correctly detected text pixels. Further, to evaluate the proposed segmentation algorithms, accuracy is also considered as performance measure. It is defined as:

$$Accuracy = CS/T \tag{5.4}$$

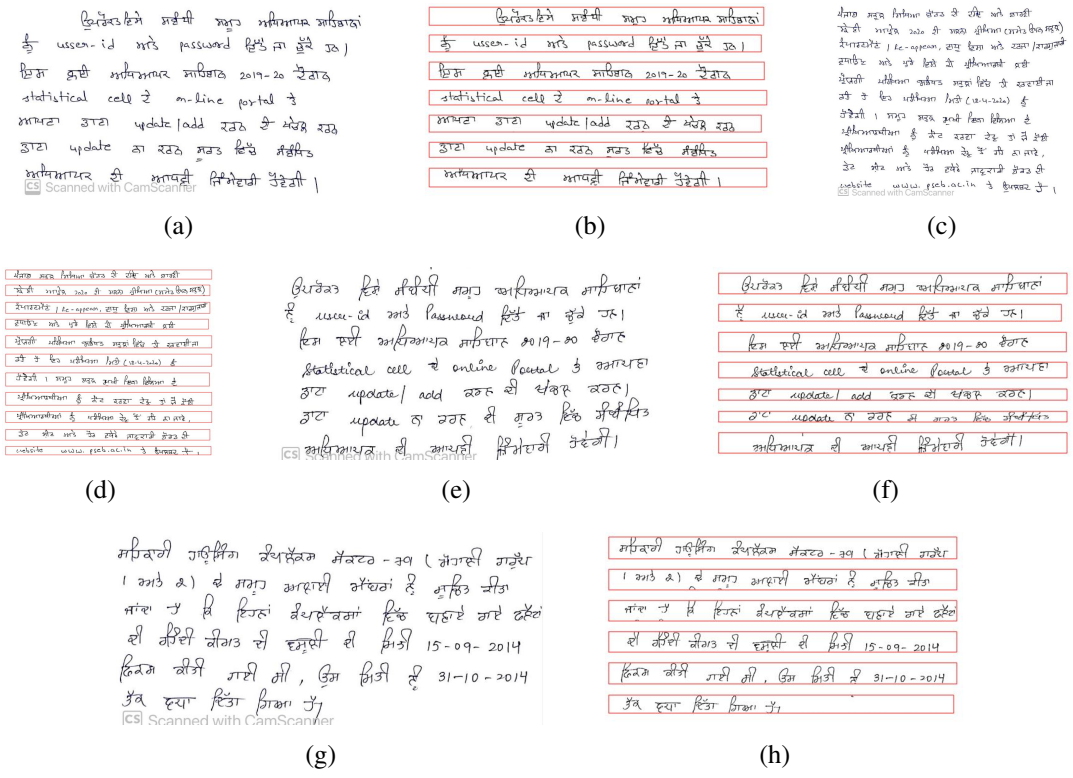
where CS is the correctly segmented data (line, word, character) and T is the total Ground truth data. RA gives more refined analysis of results as compared to accuracy as it considers correctly segmented data from the resulted data.

#### 5.1.2.1 For bilingual Handwritten document dataset

This section discusses the segmentation results using bilingual handwritten dataset of Gurumukhi-English text. It includes touching, skewed, curved and closed text lines. Similarly words with inter word and intra word gaps and touching characters have been considered.

1. For Line Segmentation: This dataset contains lesser number of straight lines as compared to IAM i.e. 287 lines. However the line segmentation accuracy of this is lower than IAM due to presence of lower and upper zone modifiers in Gurumukhi script as compared to Latin script. The merging_of_stripes algorithm increases the accuracy of straight line segmentation process. It results in accuracy of 95.8% . Some sample results are shown in Figure 5.1.

   The handwritten documents contain some skewed text lines also which is the major focus of this work. The proposed algorithm segments the skewed lines without using any skew correction algorithm. The total of 165 skewed text lines are considered in this bilingual dataset. To find the optimal cut points vertical segmentation algorithm is used. The average accuracy reported for skewed text lines is 89.75% for text with different scripts. Some results are shown Figure. 5.2. Most of the handwritten documents consist of curved text lines. Hence, in this work we have considered 185 curved text lines also. The proposed approach performes efficiently for curved text lines as



(a) (b) (c)

(d) (e) (f)

(g) (h)

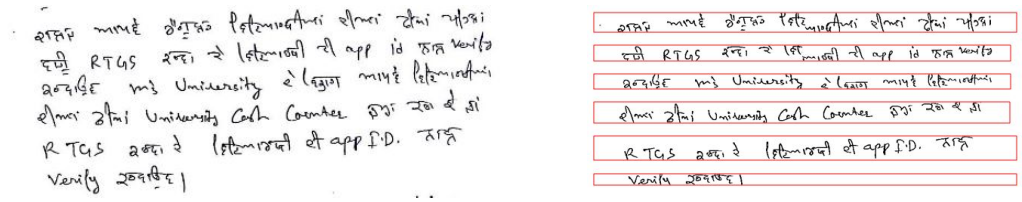
**Figure 5.1:** Results for straight lines

compared to state-of-the-art methods due to vertical segmentation. It results into the average accuracy of 96%. Some results are shown in Figure 5.3.

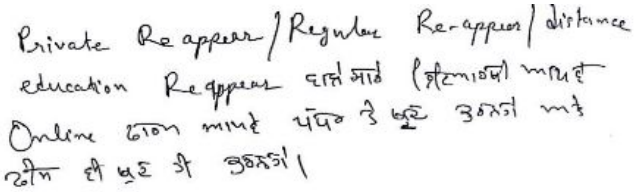
**Table 5.1:** Quantitative comparison of proposed approach with existing approaches for Line segmentation on Bilingual dataset

| Method | DR | RA | F1 measure | Accuracy | Time (in seconds) |
|---|---|---|---|---|---|
| Kundu et al. [255] | 84.62% | 85% | 84.80% | 79.24% | 195.230s |
| Ptak et al. [102] | 85.27% | 86.21% | 85.73% | 80.34% | 100s |
| Santose et al. [104] | 74.25 | 72.12 | 73.16% | 69.21% | 125s |
| Sanasam et al. [100] | 81.57% | 84.12% | 82.82% | 80.69% | 170.5s |
| Proposed | 96.95% | 95.84% | 96.39% | 92.95% | 143.488s |

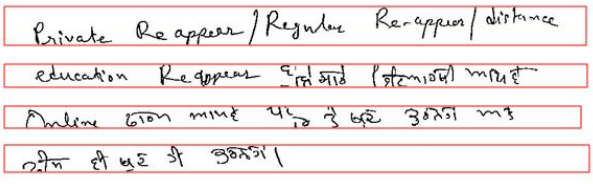
The writing style of many writers consist of closed and touching text lines. Such text lines have smaller gaps in between the lines. To process such writing styles we have considered 239 closed lines having 117 touching lines also. To segment the touching lines for scripts having modifiers present in the upper and lower zone is the real challenge. The proposed approach has achieved the accuracy of 82.9% . Some



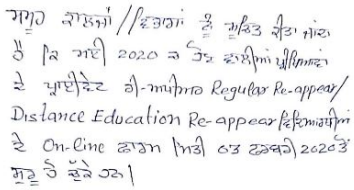
(a) (b)



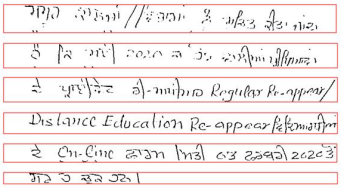
(c)



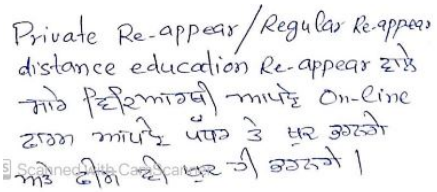
(d)



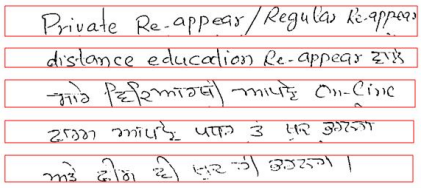
(e)



(f)

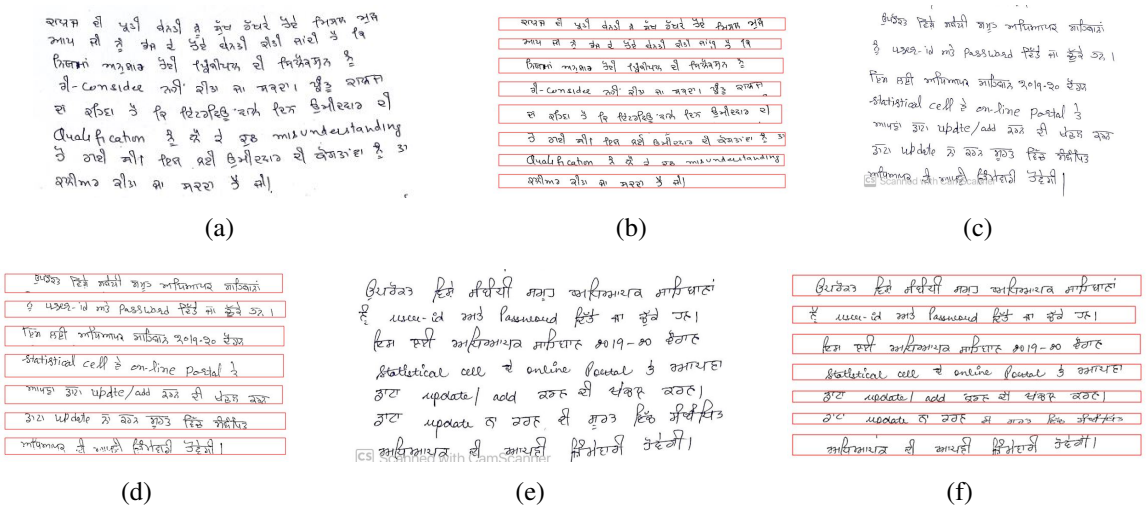


(g)



(h)

**Figure 5.2:** Results for skewed lines



(a) (b) (c)

(d) (e) (f)

**Figure 5.3:** Results for curved lines

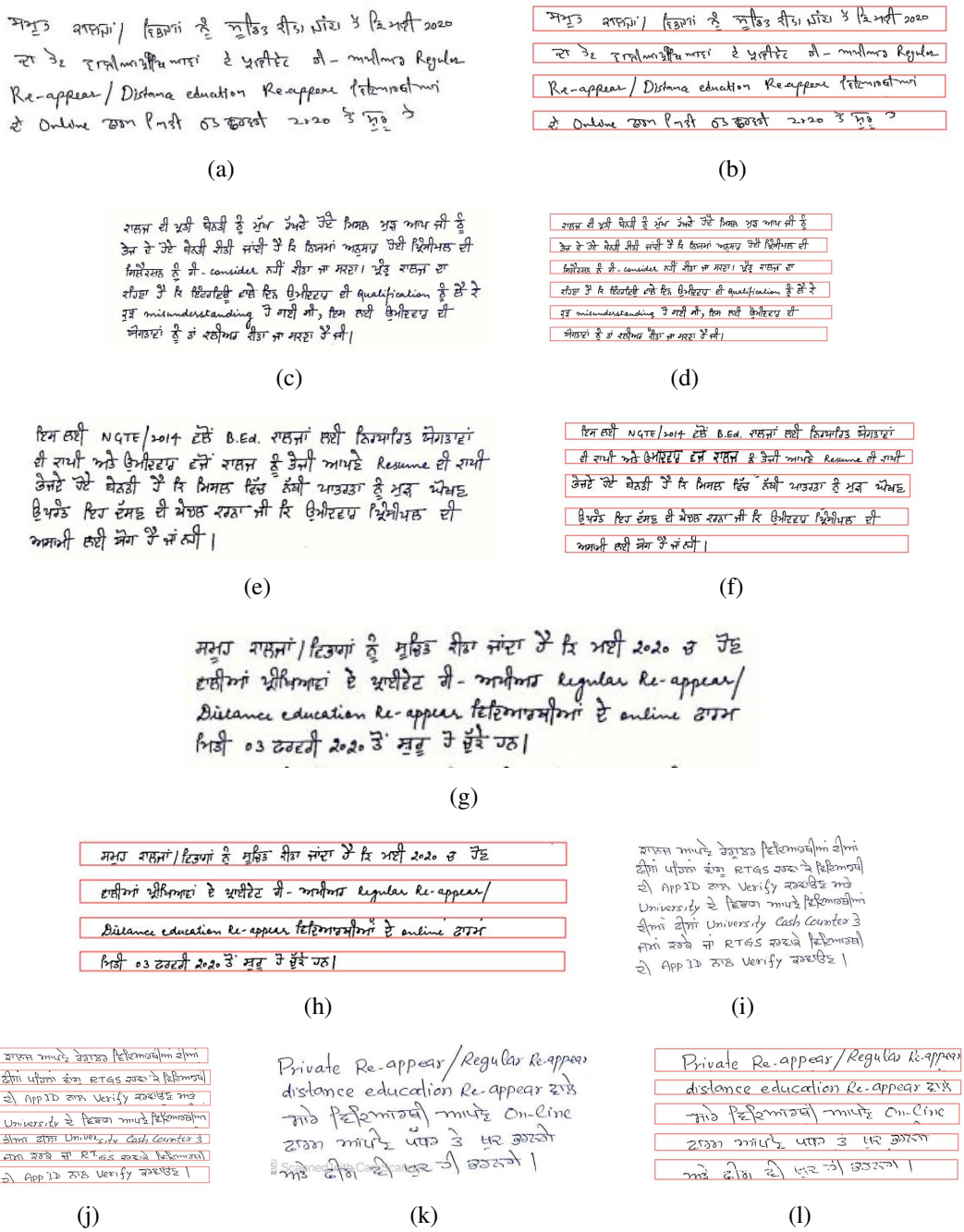
results are shown in Figure. 5.4.

2. For Word Segmentation: The proposed end point detection algorithm is able to segment the broken, cursive and skewed words in bilingual text document. The total of 6860 words have been used for segmentation. The proposed algorithm has reported an accuracy of 92.25%. The results are reported in Figure. 5.5.

**Table 5.2:** Quantitative comparison of proposed approach with existing approaches for word segmentation on Bilingual dataset

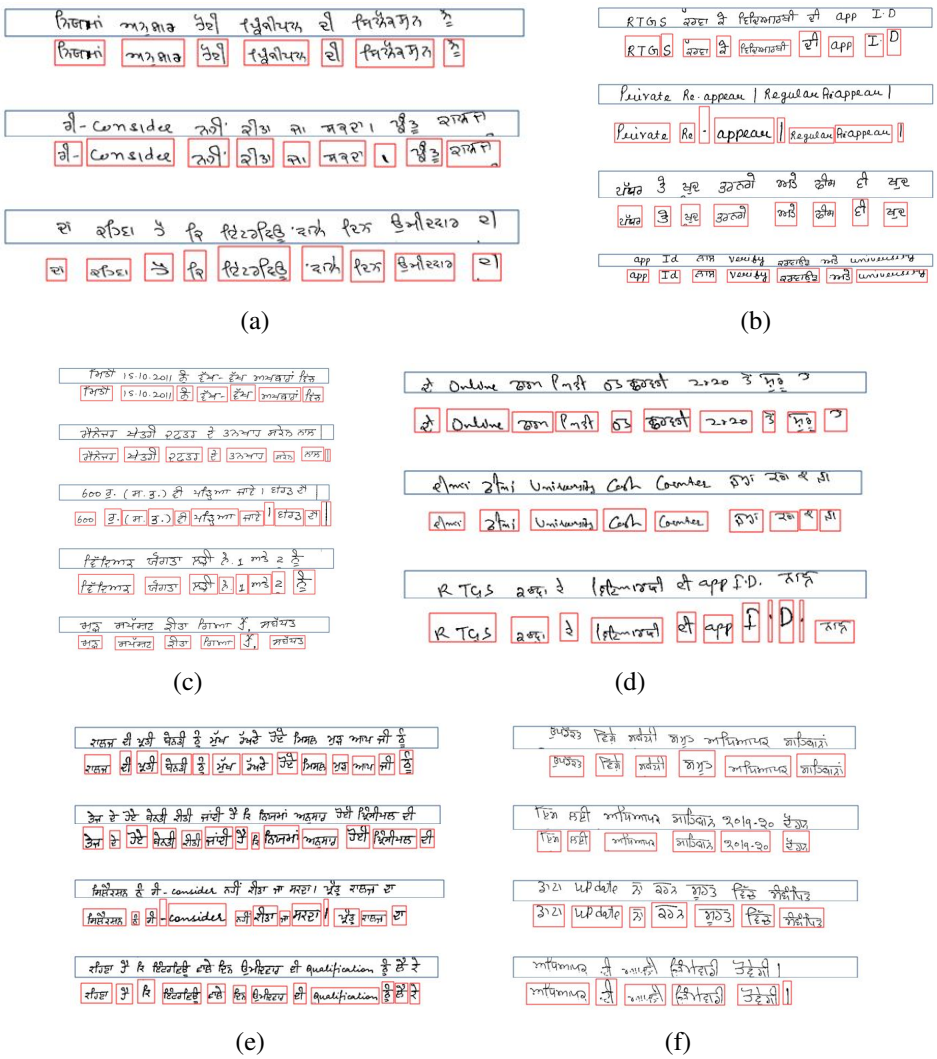
| Method | DR | RA | F1 measure | Accuracy | Time (in seconds) |
|---|---|---|---|---|---|
| Proposed | 94.05% | 96.93% | 95.46% | 92.25% | 7.630s |
| Sanasam et al. [100] | 84.2% | 82.14% | 83.15% | 81.06% | 5.4s |
| Jindal et al. [256] | 72.16 | 73.35 | 72.75% | 70.89% | 6sec |

3. **For Character Segmentation:** The character segmentation results of bilingual text are shown in Figure 5.6. This character segmentation approach uses single method to segment the text from two different scripts. It has efficiently segmented the closed, touching and skewed characters. Further, this approach can segment the characters with header lines. The average accuracy achieved for character segmentation on bilingual text is 87.25%.



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l)

**Figure 5.4:** Results for touching and closed lines



(a) (b) (c) (d) (e) (f)

**Figure 5.5:** Results for for word segmentation using end point detection algorithm

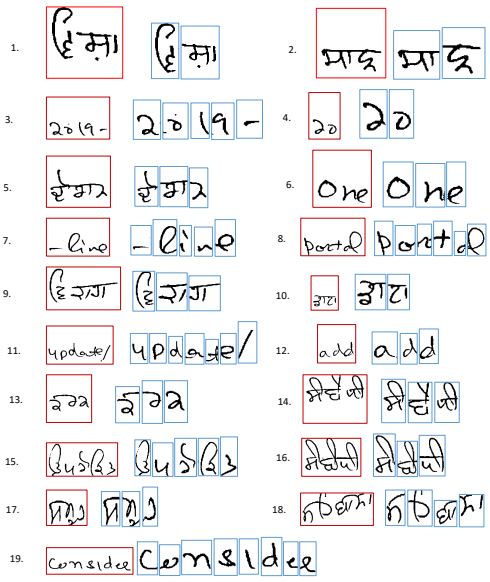


**Figure 5.6:** Character segmentation results for bilingual text

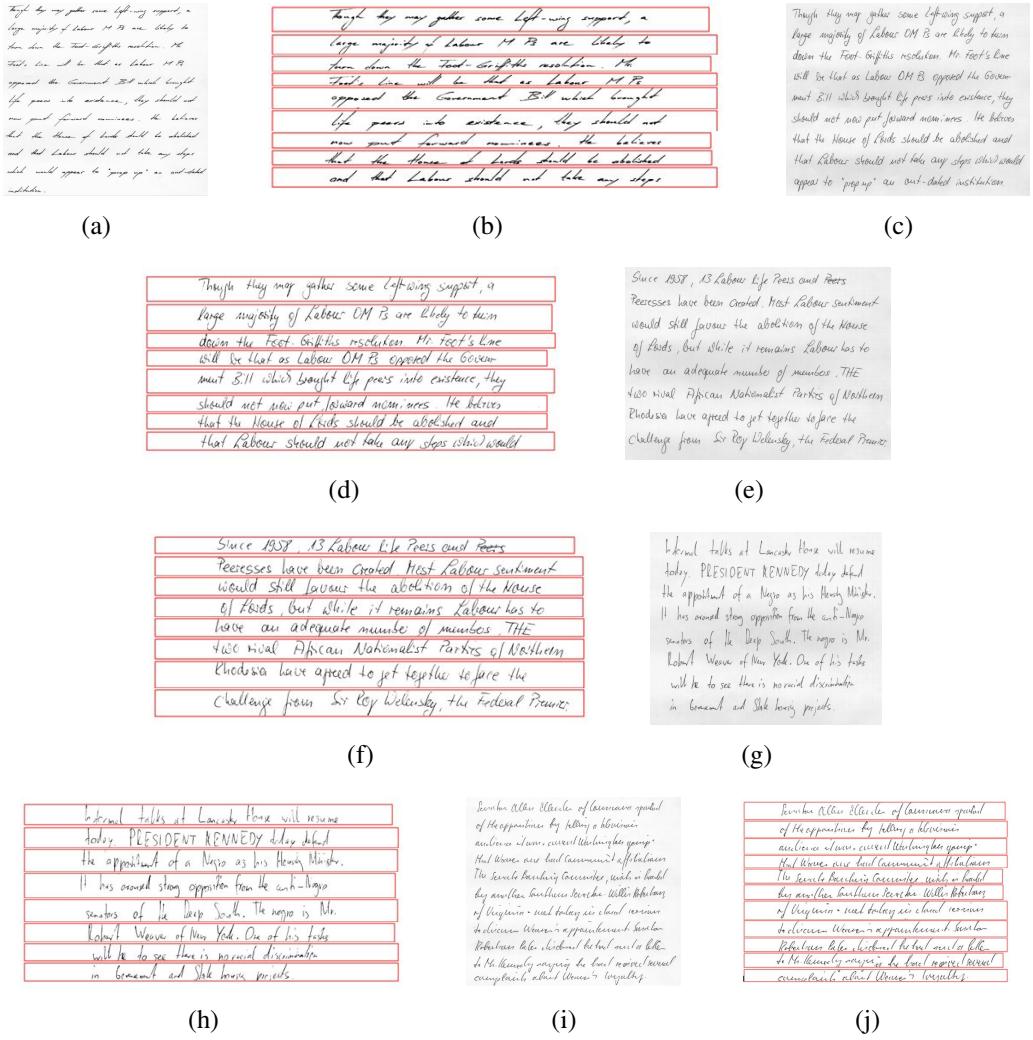
**Table 5.3:** Line Segmentation Results for Bilingual dataset

| Line type | no of correctly segmented lines | no of total lines | Accuracy proposed |
|---|---|---|---|
| Touched | 105 | 117 | 89.74% |
| Skewed | 146 | 165 | 88.48% |
| Curved | 171 | 185 | 92.43% |
| Straight | 275 | 287 | 95.81% |
| Close | 226 | 239 | 94.56 % |

#### 5.1.2.2 For IAM document dataset

The IAM dataset has 146 documents resulting into 1022 text lines. The results corresponding to line, word and character segmentation are presented as :

1. **For Line Segmentation:** IAM dataset includes skewed, straight, closed and curved



(a) (b) (c)

(d) (e)

(f) (g)

(h) (i) (j)

**Figure 5.7:** Results for Line Segmentation on IAM dataset

text lines. This work considers 146 documents resulting into 1022 lines. This dataset has less number of samples for curved text lines as compared to skewed, and touching lines. The average accuracy of 95.89% has been reported for IAM dataset. Some results are shown in Figure 5.7.

2. **For Word Segmentation:** IAM has total of 7154 words with various types like broken, cursive, and skewed. The average accuracy reported for words is 89.74%.

**Table 5.4:** Quantitative comparison of proposed approach with existing approaches for Line segmentation on IAM dataset

| Method | DR | RA | F1 measure | Accuracy | Time(in seconds) |
|---|---|---|---|---|---|
| Kundu et al. [255] | 87.62% | 90.01% | 88.79% | 84.10% | 145s |
| Ptak et al. [102] | 88.39 % | 89.10 % | 88.74% | 88% | 80s |
| Santose et al. [104] | 86.10% | 85.64% | 85.86% | 90.12% | 94s |
| Sanasam et al. [100] | 96.93% | 98.22% | 97.57% | 94.14% | 120s |
| Proposed | 96.95% | 95.84% | 96.39% | 92.95% | 143.488s |

**Table 5.5:** Quantitative comparison of proposed approach with existing approaches for word segmentation on IAM dataset

| Method | DR | RA | F1 measure | Accuracy | Time (in seconds) |
|---|---|---|---|---|---|
| Proposed | 91.27% | 88.91% | 90.07% | 89.74% | 5.8s |
| Sanasam et al. [100] | 93.56% | 91.45% | 92.49% | 90.34% | 4s |
| Jindal et al. [256] | 75.48% | 72.84% | 74.13% | 71.69% | 5sec |

**Table 5.6:** Line Segmentation Results for IAM dataset

| **Line type** | **no of correctly segmented lines** | **no of total lines** | **Accuracy proposed** |
|---|---|---|---|
| Touched | 56 | 70 | 80.0% |
| Skewed | 254 | 266 | 95.48% |
| Curved | 116 | 120 | 96.6% |
| Straight | 234 | 236 | 99.15% |
| Close | 320 | 330 | 96.6 % |

#### 5.1.2.3 Comparative analysis with state-of-the-art

This section discusses and analyses the various results achieved using proposed approach. The proposed approach has performed well for the bilingual dataset as compared to the classical segmentation approaches like projection profiles, morphological operations and connected component analysis. It can segment the text written into two different scripts without any script identification approach. Post processing phase used at each step during segmentation process improves the results. The proposed approach is able to segment the text with curved, skewed, touching, closed lines and written in different formats like application for-

mat, text started from middle of page, written in paragraphs etc. It does not require any pre determined threshold values. Hence, it can be easily adopted for text segmentation of other similar scripts like Devanagari, Bangla etc in combination with Latin script. The bilingual dataset uses handwritten samples and text content from Academic domain. Hence, it will greatly help in designing the OCR and any other text recognition task for academic domain. The quantitative comparative analysis for text line segmentation approaches has been represented in Table 5.1 and 5.4 for bilingual and IAM dataset respectively. The techniques considered in comparison for text line segmentation are based on projection profile and generative adversarial networks (GANs). Sansam et al. [100] have used modified projection profiles to find the mid point between two text lines for text line separation. This approach is unable to segment the documents where text written starts from the middle of document or from the right side of document. Further, for curved text lines separation this approach does not provide efficient solution as compared to proposed one. Moreover, it takes little more time on our dataset as compared to proposed approach due to the extra efforts needed for gap trailing between the lines. Santos et al. [104] have used morphological operations and histograms for text line separation. This is threshold dependent approach which performs efficiently for straight and closed text lines but fails to segment the curved and skewed text. Apart from these classical approaches, we have implemented another deep learning based approach for text line separation using Generative Adversarial Network (GAN) architecture. It considers the text segmentation task as image translation where a document is translated into another document with separated text lines. It requires large training data and time as compared to classical approaches to give efficient results. Similarly, Table 5.2 and 5.5 represents the quantitative comparison of proposed approach with existing approaches for word segmentation on bilingual and IAM dataset respectively. Sanasma et al. [100] have used vertical projection histograms for word segmentation where a columns with zero value is considered the boundary for word segmentation. Similarly Jindal et al. [256] have used mid point detection based approach using vertical projections for word segmentation. These approaches do not consider the intra word gap due to which it fails to segment the disconnected words correctly and results into poor word segmentation. The proposed approach uses average word size for each text line to find the inter and intra word gaps.

### 5.1.3 Script identification at character level

For script identification, exhaustive experiments have been conducted to find the most significant features and classifiers combination corresponding to our dataset. This work considers Extended-MNIST dataset [257] for English numerals and characters, while for Gurumukhi numerals and characters, dataset from Chandan et al. [258]. The bilingual dataset, designed through the use of these two datasets, contains 5 different output classes as: Gurumukhi characters (GC), English upper case characters (EC_u), English lower case characters (EC_l), Gurumukhi Numerals (GN), English Numerals (EN). These datasets include number of confusing character pairs within the datasets and between the datasets.

1. **Experiment 1: Script identification using Transfer learning and handcrafted features:** In this experiment, three major deep learning based models i.e. VGG19. ResNet50 and LeNet5 pretrained on imagenet dataset have been used. These networks are used as classifier as well as feature extractor. Further the features extracted from, deep networks are used to train traditional models i.e. SVM, RF, KNN. For classification VGG19 has achieved the highest accuracy using softmax classifier. For feature extraction, ResNet50 has outperfomed with RF classifier resulting in the accuracy of 98.01%. The CPU time taken by VGG19 and ResNet50 for SVM and RF is similar. However for KNN, VGG19 features takes little more time as compared to others. For handcrafted features, the highest accuracy has been reported with the combination of HOG features and RF classifier. The confusion matrix have been drawn to get the class wise accuracy for each model. Table 5.9 represents the results for various classifiers for script identification.

2. **Experiment 2: Combining various features for script identification:** In this research work, eleven different kinds of feature sets have been designed to analyze the combination of various features. Further three base classifiers have been used to design multiple classifiers from these feature sets. It results into 33 different classifiers. The 3 fold cross validation accuracies of models are presented in Table 5.10. From the table, it can be concluded that SVM has performed well with all the classifiers while it takes the CPU time much higher as compared to other base classifiers. It is due to

**Table 5.7:** Parameter details for training deep networks

| Network name | Learning rate | Optimizer | Epoch | Batch size | Loss function | Activation | No. of trainable parameters |
|---|---|---|---|---|---|---|---|
| LeNet5 | 0.01 | Adam | 11 | 128 | Categorical cross entropy | Tanh | 61, 281 |
| VGG19 | 0.001 | SGD | 50 | 32 | Categorical cross entropy | Relu | 21,240,010 |
| ResNet50 | 0.001 | SGD | 50 | 32 | Categorical cross entropy | Relu | 26,323,082 |

**Table 5.8:** Parameter details for training machine learning models

| Feature set | SVM (c, gamma) | RF (n_estimators) | KNN (n_estimators) |
|---|---|---|---|
| Gabor | 50, 5 | 500 | 13 |
| GLCM | 50, 5 | 1000 | 14 |
| HOG | 10, 0.01 | 1000 | 8 |
| LeNet5 | 10, 0.05 | 1000 | 5 |
| VGG19 | linear | 1000 | 6 |
| ResNet50 | linear | 1000 | 3 |

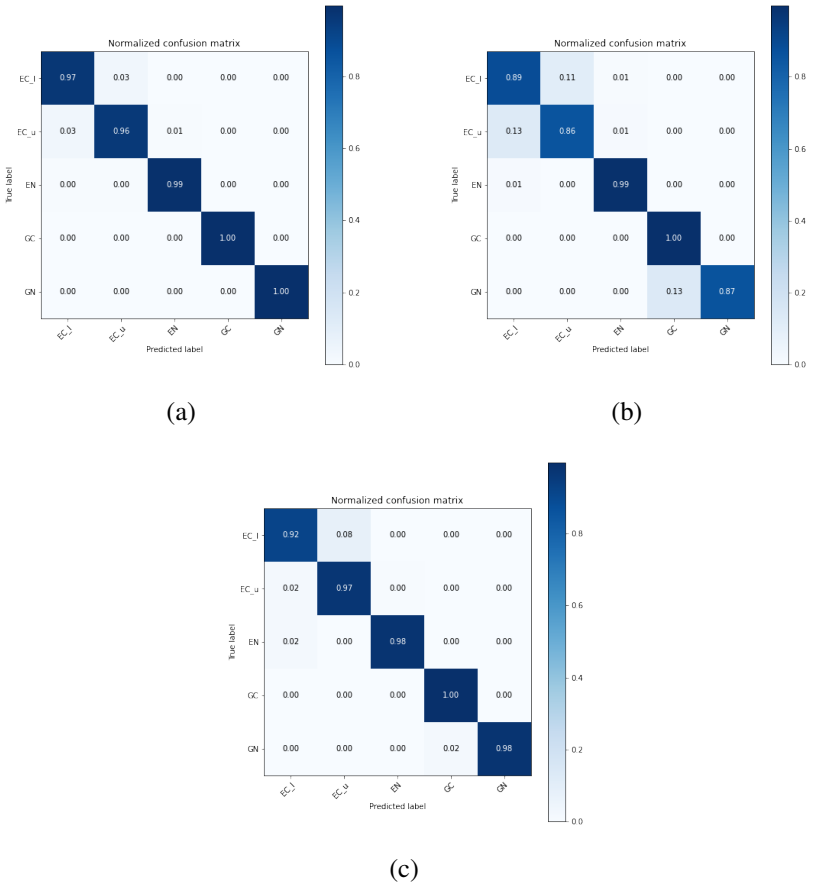



**Figure 5.8:** Confusion matrix for classification of data using: (a) VGG19; (b) LeNet5; and, (c) ResNet50

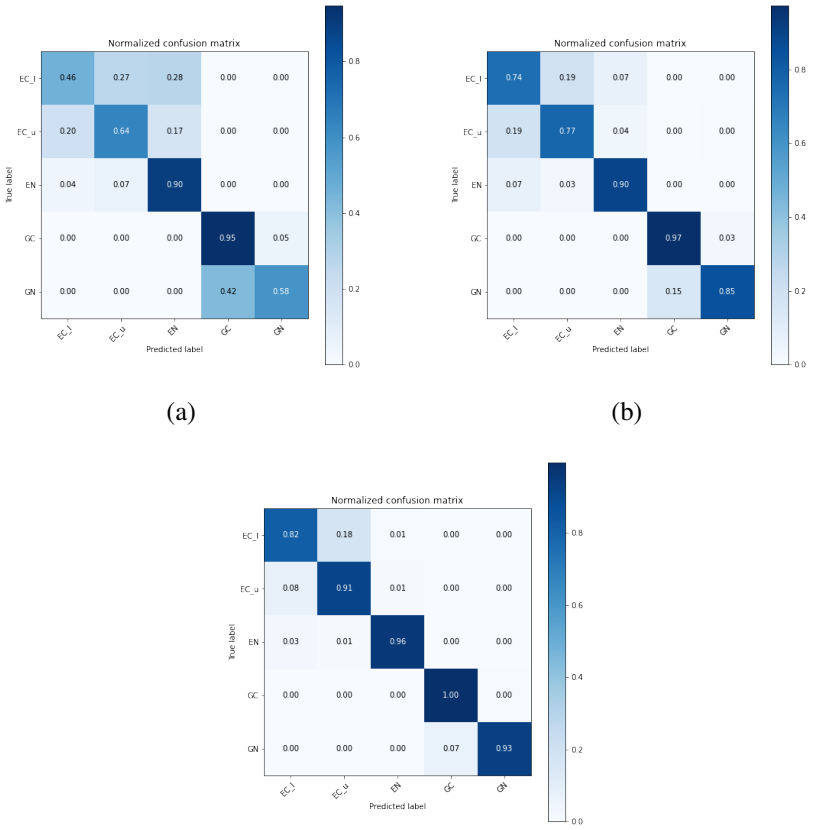



**Figure 5.9:** Confusion matrix for classification of data using, (a) Gabor features, (b) GLCM features, and, (c) HOG features

**Table 5.9:** Script recognition accuracy (in %) and CPU time (in seconds) for multiple classifiers using deep features and handcrafted features

| Feature | Feature vector | SVM | | RF | | KNN | | Softmax | |
|---|---|---|---|---|---|---|---|---|---|
| | | Accuracy | Time | Accuracy | Time | Accuracy | Time | Accuracy | Time |
| VGG19 | 25088 | 97.10 | 12257 | 97.68 | 12504 | 97.91 | 15744 | 95.88 | 12204 |
| ResNet50 | 100352 | 97.43 | 12635 | 98.01 | 12873 | 73.60 | 12610 | 95.58 | 12610 |
| LeNet5 | 400 | 94.37 | 633 | 93.22 | 110 | 92.45 | 250 | 92.30 | 120 |
| Gabor (F1) | 64 | 78.00 | 1827 | 77.00 | 1585 | 75.35 | 1645 | - | - |
| GLCM (F2) | 48 | 86.00 | 2166 | 86.00 | 1589 | 82.94 | 1644 | - | - |
| HOG (F3) | 2400 | 93.28 | 644 | 94.45 | 330 | 92.63 | 500 | - | - |

the parameter optimization in SVM. KNN has performed lowest as compared to SVM and RF.

**Table 5.10:** Script recognition accuracy (in %) and CPU time (in seconds) for multiple classifiers with combination of different feature sets

| Feature | Feature vector | SVM | | RF | | KNN | |
|---|---|---|---|---|---|---|---|
| | | Accuracy | Time | Accuracy | Time | Accuracy | Time |
| LeNet+Gabor | 464 | 92.61 | 3610 | 90.21 | 1950 | 92.62 | 1973 |
| LeNet+GLCM | 484 | 94.26 | 3611 | 91.83 | 2731 | 93.20 | 1971 |
| LeNet+HOG | 2800 | 93.22 | 17720 | 94.14 | 349 | 92.73 | 3134 |
| F2+F3 | 2448 | 93.93 | 16897 | 94.53 | 1814 | 92.95 | 4158 |
| F1+F3 | 2464 | 93.73 | 17068 | 94.14 | 1946 | 92.91 | 2057 |
| F1+F2 | 112 | 87.49 | 3529 | 85.28 | 1844 | 83.63 | 1845 |
| F1+F2+F3 | 2512 | 93.93 | 18890 | 94.37 | 3379 | 92.97 | 5523 |
| F1+F2+F4 | 512 | 94.57 | 21880 | 91.83 | 3231 | 93.11 | 3235 |
| F1+F2+F3+F4 | 2912 | 94.04 | 21440 | 94.06 | 3476 | 92.97 | 3544 |
| F1+F3+F4 | 2864 | 93.95 | 16792 | 93.83 | 1911 | 92.91 | 2014 |
| F2+F3+F4 | 2848 | 94.01 | 19580 | 94.11 | 1909 | 93.12 | 2011 |

3. **Experiment 3: Script identification using Ensemble approaches:** This experiment considers three kinds of ensemble approaches viz., Voting, Bagging, and boosting. Voting based ensemble classifier combines the complimentary nature of different classifiers. Table 5.11 represents the accuracies for various feature sets with voting based

classifier for script identification. Gabor and HOG features have reported highest accuracies for voting based classifier. The best combination of features are LeNet-GLCM (95.53%), LeNet-HOG (95.68%), GLCM-HOG (95.83%), Gabor-HOG (95.99%), Gabor-GLCM-HOG (95.76%), Gabor-GLCM-LeNet (95.45%), Gabor-GLCM-HOG-LeNet (95.76%), Gabor-HOG-LeNet (95.83%) and GLCM-HOG-LeNet (95.83%). For deep networks the accuracy reported for voting based classifiers is lesser as compared to individual classifier. Hence for ensemble approaches we need weak base classifiers. There is not much rise in CPU time for voting based classifiers with respect to individual classifiers.

**Table 5.11:** Script recognition accuracy (in %) and CPU time (in seconds) using voting based classifier

| Feature | Feature vector | SVM | RF | KNN | Voting based | Time |
|---|---|---|---|---|---|---|
| Gabor(F1) | 64 | 78.00 | 77.00 | 75.35 | 75.19 | 1603 |
| GLCM (F2) | 48 | 86.00 | 86.00 | 82.94 | 87.59 | 1708 |
| HOG(F3) | 2400 | 93.28 | 94.45 | 92.63 | 95.14 | 2318 |
| LeNet(F4) | 400 | 94.69 | 93.22 | 92.68 | 92.75 | 532 |
| LeNet+Gabor | 464 | 92.61 | 90.21 | 92.62 | 93.83 | 1950 |
| LeNet+GLCM | 484 | 94.26 | 91.83 | 93.20 | 95.53 | 1851 |
| LeNet+HOG | 2800 | 93.22 | 94.14 | 92.73 | 95.68 | 829 |
| F2+F3 | 2448 | 93.93 | 94.53 | 92.95 | 95.83 | 2195 |
| F1+F3 | 2464 | 93.73 | 94.14 | 92.91 | 95.99 | 2639 |
| F1+F2 | 112 | 87.49 | 85.28 | 83.63 | 86.97 | 3504 |
| F1+F2+F3 | 2512 | 93.93 | 94.37 | 92.97 | 95.76 | 3742 |
| F1+F2+F4 | 512 | 94.57 | 91.83 | 93.11 | 95.45 | 3606 |
| F1+F2+F3+F4 | 2912 | 94.04 | 94.06 | 92.97 | 95.76 | 4093 |
| F1+F3+F4 | 2864 | 93.95 | 93.83 | 92.91 | 95.83 | 2511 |
| F2+F3+F4 | 2848 | 94.01 | 94.11 | 93.12 | 95.83 | 2521 |
| VGG19 | 25088 | 97.10 | 97.68 | 97.91 | 95.27 | 15 hours |
| ResNet50 | 100352 | 97.43 | 98.01 | 73.60 | 94.96 | 14 hours |

Further, from ensemble approaches Bagging based ensemble approach has used three base classifiers i.e. SVM, KNN, and RF. For traditional feature sets KNN has performed well as compared to other for classification. KNN individually has not performed well, as it is considered as weak classifier. However with ensemble the per-

formance of KNN has improved significantly. Some of the fine accuracies achieved by KNN as base classifier with the best performing feature sets are: Gabor-HOG (93.23%), HOG (93.30%), Gabor-GLCM-HOG (93.36%), LeNet-GLCM (93.92%), LeNet-HOG (93.54%), and Gabor-HOG-LeNet (93.35%) (Table 5.12). Here, again, HOG takes more time for classification as compared to Gabor and GLCM. For pre-trained models, like ResNet50 has reported good accuracy with SVM. However, it takes a lot more time as compared to RF.

The third i.e., Boosting based ensemble approach uses Adaboost classifier. It uses RF, decision tree and SVM as base classifiers. Adaboost has performed well with HOG features and RF base classifiers. For pre-trained models, Adaboost has achieved the maximum accuracy of 98.43% with the combination of ResNet50 and SVM. However, this improvement in accuracy increases the training cost of network in terms of CPU time as compared to RF and decision tree.

**Table 5.12:** Script recognition accuracy (in %) and CPU time (in seconds) using bagging based ensemble classifier

| Feature | Feature vector | SVM | KNN | RF | Time |
|---|---|---|---|---|---|
| Gabor(F1) | 64 | 77.02 | 67.62 | 74.51 | 1620 |
| GLCM (F2) | 48 | 75.33 | 83.53 | 83.16 | 1615 |
| HOG(F3) | 2400 | 84.80 | 93.30 | 91.88 | 4799 |
| LeNet(F4) | 400 | 88.28 | 92.45 | 90.59 | 562 |
| LeNet+Gabor | 464 | 67.24 | 92.20 | 88.35 | 2140 |
| LeNet+GLCM | 484 | 76.85 | 93.92 | 89.71 | 2532 |
| LeNet+HOG | 2800 | 85.36 | 93.54 | 92.06 | 2813 |
| F2+F3 | 2448 | 84.10 | 92.90 | 92.48 | 3676 |
| F1+F3 | 2464 | 84.08 | 93.23 | 92.13 | 8990 |
| F1+F2 | 112 | 74.63 | 84.00 | 83.20 | 3758 |
| F1+F2+F3 | 2512 | 85.55 | 93.36 | 92.33 | 15477 |
| F1+F2+F4 | 512 | 72.48 | 92.81 | 89.79 | 3668 |
| F1+F2+F3+F4 | 2912 | 84.33 | 92.87 | 91.17 | 4273 |
| F1+F3+F4 | 2864 | 83.70 | 93.35 | 92.01 | 3694 |
| F2+F3+F4 | 2848 | 83.84 | 92.91 | 92.21 | 4592 |
| VGG19 | 25088 | 98.61 | 97.76 | 98.16 | 15148 |
| ResNet50 | 100352 | 98.82 | 70.50 | 98.57 | 25680 |

**Table 5.13:** Script recognition accuracy (in %) and CPU time (in seconds) using boosting based ensemble classifier

| Feature | Feature vector | SVM | RF | Decision Tree | Time |
|---|---|---|---|---|---|
| Gabor(F1) | 64 | 49.15 | 76.34 | 68.56 | 2238 |
| GLCM (F2) | 48 | 52.54 | 83.97 | 69.26 | 1572 |
| HOG(F3) | 2400 | 65.87 | 92.44 | 68.87 | 374 |
| LeNet(F4) | 400 | 59.01 | 90.60 | 31.04 | 135 |
| LeNet+Gabor | 464 | 52.91 | 85.82 | 68.25 | 1781 |
| LeNet+GLCM | 484 | 53.01 | 87.75 | 68.56 | 1724 |
| LeNet+HOG | 2800 | 63.11 | 91.91 | 48.68 | 475 |
| F2+F3 | 2448 | 52.96 | 87.15 | 71.18 | 1925 |
| F1+F3 | 2464 | 58.22 | 90.83 | 69.02 | 1935 |
| F1+F2 | 112 | 51.55 | 82.89 | 68.25 | 3504 |
| F1+F2+F3 | 2512 | 53.88 | 91.37 | 68.33 | 3504 |
| F1+F2+F4 | 512 | 52.45 | 90.21 | 67.41 | 3259 |
| F1+F2+F3+F4 | 2912 | 53.77 | 91.67 | 70.10 | 3560 |
| F1+F3+F4 | 2864 | 52.78 | 91.83 | 71.64 | 1987 |
| F2+F3+F4 | 2848 | 62.18 | 92.37 | 70.03 | 1988 |
| VGG19 | 25088 | 96.12 | 97.27 | 75.57 | 13,208 |
| ResNet50 | 100352 | 98.43 | 97.60 | 82.24 | 15561 |

### 5.1.4 OCR Results

After script identification, the next step is to recognize the character corresponding to script which requires OCR. Three kinds of OCR corresponding to three major scripts i.e. English, Gurumukhi and Alphanumeric have been designed. To evaluate the proposed OCRs many performance measures apart from accuracy have been used like: precision, recall, F1 score, CPU time, validation and training loss. These OCRs are designed using handcrafted and deep learning features and classifiers.

#### 5.1.4.1 Gurumukhi OCR

This section evaluates the two different kinds of OCRs for Gurumukhi based upon the dataset i.e. Presegmented characters and composite characters.

1. **Pre-segmented Characters:** The presegmented dataset has 50 classes of characters and matras. The traditional feature set is able to recognize this dataset in lesser training

time Table . From tradtional features, HOG feature set has performed well while, from deep learning ResNet50 has achieved highest accuracy.

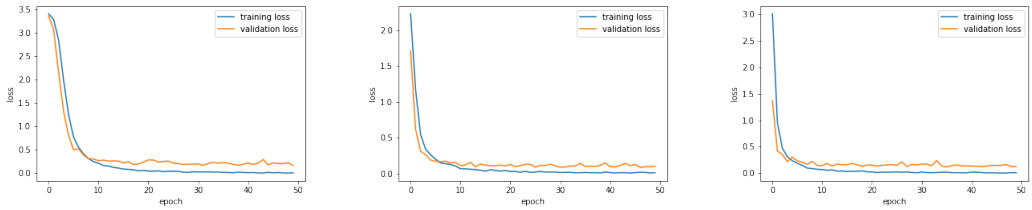
**Table 5.14:** Results for Presegmented Data

| Index | Type | SVM | | | | RF | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Precision | Recall | F1 | Accuracy | Precision | Recall | F1 | Accuracy |
| F1 | Gabor | 43 | 42 | .41 | 41.98 | 42 | 42 | .41 | 41.59 |
| F2 | GLCM | 97 | 96 | .97 | 94.78 | 93 | 92 | .92 | 91.91 |
| F3 | HOG | 97 | 96 | .97 | 96.47 | 93 | 92 | .92 | 91.91 |
| F4 | Vgg19 | 99 | 99 | .99 | 98.95 | 99 | 99 | .99 | 98.82 |
| F5 | ResNet50 | 100 | 99 | .99 | 99.47 | 99 | 99 | .99 | 99.34 |
| F6 | LeNet5 | 95 | 95 | .95 | 94.78 | 94 | 93 | .93 | 93.08 |

2. **Composite Characters:** Composite dataset has two level classification one for characters and other for matras. Here, deep neural networks have performed well in classifying the complex data. This dataset has greater similarity between the classes as compared to presegmented dataset. From deep networks, ResNet50 has performed well at both levels of classification for composite CR. The CPU time needed to train the composite dataset is two times higher than presegmented data. However once this training is done, the time for testing the results is almost similar to presegmented data. The training and validation losses for various deep learning models trained on composite and presegmented dataset are shown in Figures 5.10, 5.11, and 5.12.

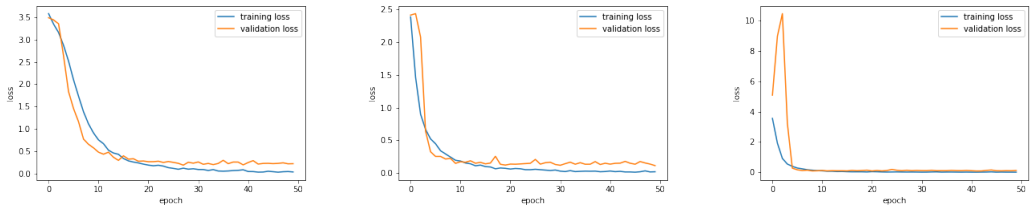
**Table 5.15:** Results for Composite Data Level1

| Index | Type | SVM | | | | RF | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Precision | Recall | F1 | Accuracy | Precision | Recall | F1 | Accuracy |
| F1 | Gabor | 23 | 22 | .22 | 16.26 | 24 | 23 | .23 | 22.85 |
| F2 | GLCM | 43 | 43 | .42 | 43.37 | 37 | 37 | 37 | 37.42 |
| F3 | HOG | 56 | 56 | .55 | 56.23 | 51 | 51 | .50 | 51.36 |
| F4 | Vgg19 | 99 | 99 | .99 | 98.72 | 98 | 98 | .98 | 98.29 |
| F5 | ResNet50 | 99 | 99 | .99 | 98.86 | 98 | 98 | .98 | 98.43 |
| F6 | LeNet5 | 20 | 19 | .19 | 18.89 | 26 | 23 | .23 | 22.58 |



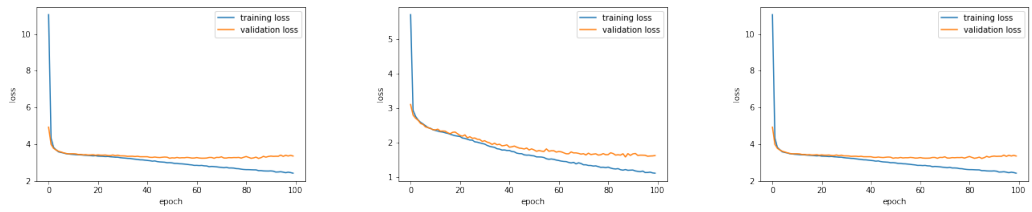

(a) (b) (c)

**Figure 5.10:** Training and Validation Loss for VGG19 with: (a) Composite Level1 ; (b) Composite Level2 ; (c) Presegmented;




(a) (b) (c)

**Figure 5.11:** Training and Validation Loss for ResNet50 with: (a) Composite Level1 ; (b) Composite Level2 ; (c) Presegmented;




(a) (b) (c)

**Figure 5.12:** Training and Validation Loss for LeNet5 with: (a) Composite Level1 ; (b) Composite Level2 ; (c) Presegmented;

**Table 5.16:** Results for Composite Data Level2

| Index | Type | SVM | | | | RF | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Precision | Recall | F1 | Accuracy | Precision | Recall | F1 | Accuracy |
| F1 | Gabor | 43 | 42 | .41 | 22.11 | 42 | 42 | .41 | 41.45 |
| F2 | GLCM | 62 | 62 | .62 | 62.32 | 57 | 56 | .56 | 56.31 |
| F3 | HOG | 64 | 64 | .63 | 64.47 | 62 | 62 | .61 | 62.14 |
| F4 | Vgg19 | 99 | 99 | .99 | 98.57 | 98 | 98 | .98 | 98.0 |
| F5 | ResNet50 | 99 | 99 | .99 | 99.14 | 99 | 99 | .99 | 98.71 |
| F6 | LeNet5 | 49 | 48 | .48 | 48.42 | 57 | 58 | .56 | 57.85 |

#### 5.1.4.2 English OCR

The dataset for English characters have 52 classes containing small and capital letters. Data augmentation have been used to enlarge the dataset for training of deep neural networks. Dataset is divided into training, validation, and testing dataset.

**Table 5.17:** Results for English Data

| Index | Type | SVM | | | | RF | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Precision | Recall | F1 | Accuracy | Precision | Recall | F1 | Accuracy |
| F1 | Gabor | 17 | 16 | .16 | 16.11 | 18 | 16 | .16 | 15.5 |
| F2 | GLCM | 56 | 53 | .53 | 53.37 | 42 | 42 | .41 | 41.98 |
| F3 | HOG | 97 | 96 | .96 | 74.05 | 93 | 92 | .92 | 91.91 |
| F4 | Vgg19 | 91 | 90 | .90 | 90.08 | 91 | 91 | .90 | 90.71 |
| F5 | ResNet50 | 94 | 94 | .94 | 93.67 | 95 | 94 | .94 | 94.30 |
| F6 | LeNet5 | 73 | 70 | .70 | 70.46 | 73 | 70 | .69 | 70.04 |

#### 5.1.4.3 Alphanumeric Data

Alphanumeric dataset contains data with 22 classes to recognize the numbers and special characters.

## 5.2 Results Analysis for Gurumukhi-English script

The results mentioned in chapter 5 depicts the efficiency of proposed methods to handle various challenges of bilingual handwriting recognition. The handwritten documents contain

**Table 5.18:** Results for Alphanumeric Data

| Index | Type | SVM | | | | RF | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Precision | Recall | F1 | Accuracy | Precision | Recall | F1 | Accuracy |
| F1 | Gabor | 51 | 48 | .49 | 48.33 | 45 | 43 | .43 | 43.33 |
| F2 | GLCM | 87 | 88 | .87 | 87.5 | 86 | 85 | .86 | 86.66 |
| F3 | HOG | 76 | 74 | .74 | 76.33 | 71 | 70 | .70 | 70.04 |
| F4 | Vgg19 | 97 | 96 | .96 | 97.8 | 96 | 97 | .96 | 96.71 |
| F5 | ResNet50 | 98 | 96 | .96 | 98.67 | 97 | 96 | .96 | 97.30 |
| F6 | LeNet5 | 91 | 90 | .90 | 90.46 | 89 | 90 | .89 | 89.45 |

some skewed text lines also which is the major focus of this work. The proposed algorithm segments the skewed lines without using any skew correction algorithm. The total of 165 skewed text lines are considered in this bilingual dataset. To find the optimal cut points vertical segmentation algorithm is used. The average accuracy reported for skewed text lines is 89.75% for text with different scripts.

The writing style of many writers consist of closed and touching text lines. Such text lines have smaller gaps in between the lines. To process such writing styles we have considered 239 closed lines having 117 touching lines also. To segment the touching lines for scripts having modifiers present in the upper and lower zone is the real challenge. The proposed approach has achieved the accuracy of 82.9%.

The proposed end point detection algorithm is able to segment the broken, cursive and skewed words in bilingual text document. The total of 6860 words have been used for segmentation. The proposed algorithm has reported an accuracy of 92.25%.

This character segmentation approach uses single method to segment the text from two different scripts. It has efficiently segmented the closed, touching and skewed characters. Further, this approach can segment the characters with header lines. The average accuracy achieved for character segmentation on bilingual text is 87.25%.

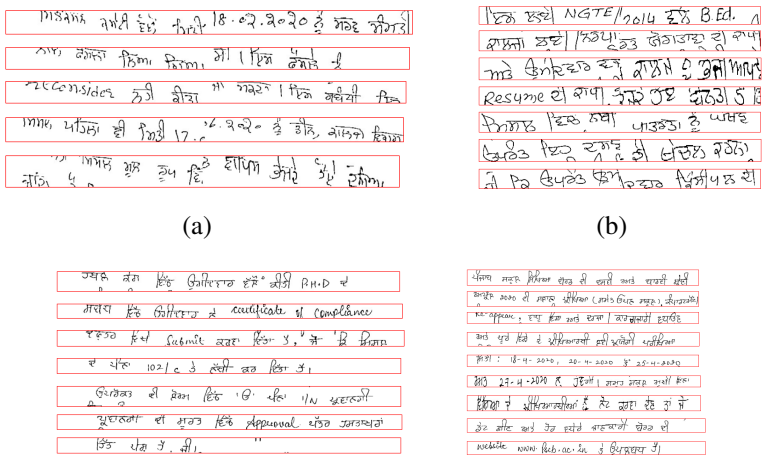
Voting based ensemble classifier combines the complimentary nature of different classifiers. Table 5.11 represents the accuracies for various feature sets with voting based classifier for script identification. Gabor and HOG features have reported highest accuracies for voting based classifier.For deep networks the accuracy reported for voting based classifiers is lesser as compared to individual classifier. Hence for ensemble approaches we need weak base

classifiers.

Composite dataset has two level classification one for characters and other for matras. Here, deep neural networks have performed well in classifying the complex data. This dataset has greater similarity between the classes as compared to presegmented dataset. From deep networks, ResNet50 has performed well at both levels of classification for composite CR. The CPU time needed to train the composite dataset is two times higher than presegmented data. However once this training is done, the time for testing the results is almost similar to presegmented data.

## 5.3 Error Analysis

This section is dedicated to error analysis, discussing common misclassifications and potential reasons behind these errors. Figure 5.13 represents some of the common misclassification errors for line segmentation. Fig 5.13(a) shows the text sample with curved skew and low quality text. The results for this are not very efficient as in many lines the text received is not correct and it is over segmented and undersegmented. The potential reason behind this is that our approach uses projection profiles method which sometime lacks in segmenting the curved text. In Fig 5.13(b) we have sample which includes text with closed, curved, touching content. In this again we have oversegmentation and undersegmentation errors. The other errors which are not correctly segmented from the segmented text as in second line third word and similarly in line 3 and 4. In the segmentation process to segment the touching and closed text when we do partitioning and merging process, that results into such errors. Fig 5.13(c) shows sample where the segmentation process missed out some content. Fig 5.14 represents the errors during word segmentation. The major reason for such errors (Fig. 5.14(b), 5.14(b), 5.14(b)) are vertical segmentation during line segmentation phase which leads to such errors when we merge two stripes again. Some errors like (Fig. 5.14(c), 5.14(e)) are due to low quality of scanning process. Further, Fig. 5.14(g) represents the undersegmentation error where proposed method fails for disconnected words. Fig 5.15 represents the errors during script identification and composite character recognition. These are interrelated as the errors of script identification process will effect the error of character recognition. As for



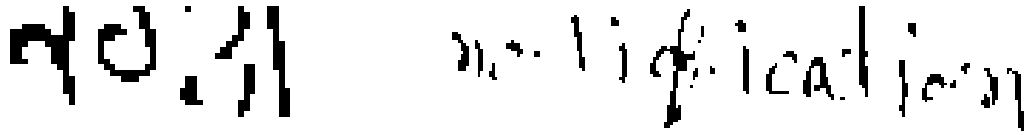
(a) (b) (c) (d)

**Figure 5.13:** Errors for line segmentations

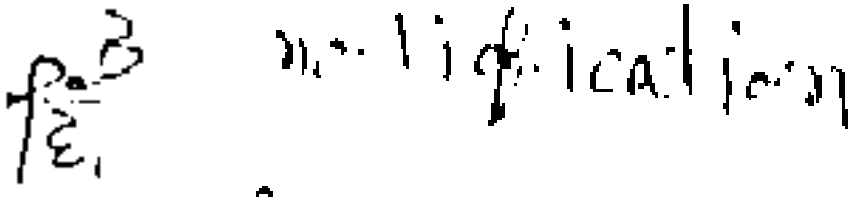
Fig. 5.14(a) we have got the right answer for script identification (Fig. 5.14(b) ) but it was unable to get the composite character recognition. Although the matra was identified correct but character identified was wrong. The reason for this is the distorted shape of the input character. Similar case is with Fig. 5.14(g) where the input character was distorted, hence it results into wrong script classification.

The performance of proposed system greatly influenced by the writing styles, camera quality and background of writers. Hence, in our dataset we have included samples with large variety including background of users. As in Fig. 5.13, 5.14, and 5.15 we have seen that the variation in handwriting as well as quality of text highly impact the recognition process. In Fig. 5.15(a) the script identification results are correct even though the quality of image and handwriting was not good but for character recognition process it gives errors. Similar, is the case with Fig. 5.15(f) and Fig 5.15(g). Further, Fig 5.13(a), 5.13(b) and 5.13(c), 5.13 (d) represents the results of writers with different backgrounds and age and paper quality. Table 5.3 and 5.6 represents the performance of the system corresponding to different kinds of handwriting styles.

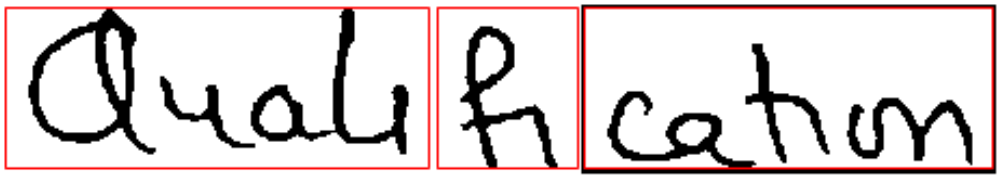
(a)



(b) (c)



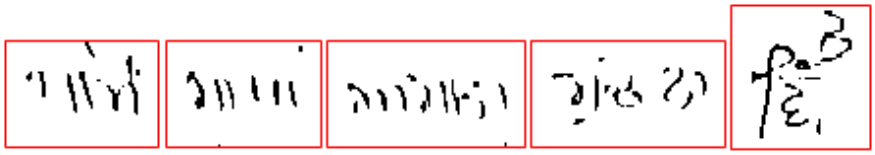
(d) (e)



(f)

(g)

(h)

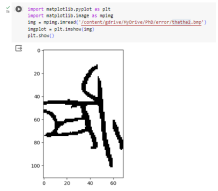


(i)

**Figure 5.14:** Errors for word segmentation

(a)

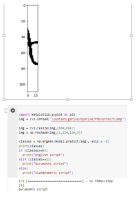

(b)



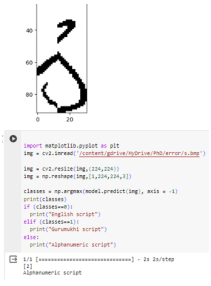
(c)


(d)

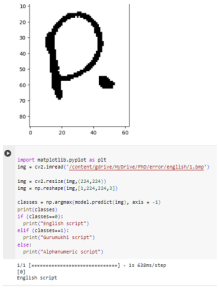

(e)



(f)



(g)



(h)

**Figure 5.15:** Errors for classification

## 5.4 Case Study: Testing BHTRforAD framework

The proposed BHTRforAD system is designed to recognize the bilingual text for handwritten documents from academic domain. The proposed system processes handwritten documents and recognizes the bilingual text. The GUI for the text recognition system has been designed using matlab. The details related to the design principles, usability considerations, and how the GUI facilitates interaction with the system are as follow:
Some of the design principles considered are as :

1. Visibility: In the designed system we have taken care of the visibility principle where users are able to see the options and features available to them without having to search extensively. This includes clear labeling, visible icons, and easily accessible menus.

2. Consistency: The designed system follows the consistency as the elements like upload image, display output within the interface have and look the same across different screens and sections. Consistency in layout, design, and interaction patterns helps users navigate the interface more easily.

3. Simplicity: We have tried to keep the interface simple and avoid clutter. We have removed unnecessary elements and features that might overwhelm or confuse users.

Some of the usability considerations are as:

1. Feedback and Error Prevention: We have designed the interface to prevent errors whenever possible. But in future our target will be to use descriptive error messages and provide clear instructions on how to correct them.

2. User-Centered Design: We tried to design the GUI with the end user in mind by understanding their needs, preferences, and skill levels.

3. Efficiency: We have optimized the interface for efficiency by reducing the number of steps required to complete common tasks.

4. Visibility of System Status: The output has been displayed of each process which provides feedback to users about what is happening within the system. We have used

status messages, and drop down menus to see the outputs which keeps users informed about the current state of the interface and any ongoing processes.

### 5.4.1 Test case execution

The test cases are developed to check the various phases of bilingual handwritten text recognition. Numerous test-cases are executed corresponding to each phase and which get failed are worked upon again. Passed test-cases are presented in the Table 5.19. Test case 1 is executed to check the line segmentation of handwritten document. In Figure 5.16, The test is passed. For testing the skewed text line segmentation, test case 2 has been executed. It is shown in the Figure 5.17. So, the document with skewed text is given as input to the segmentation module. The output of segmented text lines is similar to the expected output. Hence, the test is passed. Test case 3 is designed to test the word segmentation where text lines are given as input to the word segmentation module. The output as segmented words is obtained, which is the expected output. Figure 5.18 shows test case 3 is passed. Further lines having words with some inter gaps are given to the system in test case 4. The output of this test results into exact required words as shown in Figure 5.19. Hence, this test is passed. The next test i.e. test case 5 is designed for characters segmentation where the segmented words are used as input. The expected output of test case is the segmented characters. Figure 5.20 shows output of the test case which is similar to expected. To identify the script of segmented characters, test case 6 is executed for script identification. The output of this test case is the required script for text. The output obtained of this test is the required script of text which is equal to the expected output. Hence, the test is passed.

**Table 5.19:** Test cases.

| Test ID | Test Objective | Test Procedure | Expected Result | Actual Result | Status |
|---|---|---|---|---|---|

Continued on next page

**Table 5.19 – continued from previous page**

| Test ID | Test Objective | Test Procedure | Expected Result | Actual Result | Status |
|---|---|---|---|---|---|
| 1 | To test the line segmentation | Enter the handwritten document | output: segmented text lines | output: segmented text lines | PASS |
| 2 | To test skewed lines segmentation | identify the skewed text lines | output: segmented text lines | output: segmented text lines | PASS |
| 3 | To test word segmentation | Enter segmented text lines | output: segmented words | output: segmented words | PASS |
| 4 | To test word segmentation of words with inter gap | Enter segmented text lines | output: segmented words | output: segmented words | PASS |
| 5 | To test character segmentation | Enter segmented words | output: segmented characters | output: segmented characters | PASS |
| 6 | To test script identification | Enter segmented characters | output: script class | output: script class | PASS |
| 7 | To test character recognition | Select the character recognizer corresponding to script class | Output: character class | Output: character class | PASS |
| 8 | To test the generation of doc file | Write the recognized text into doc file | Output: read the doc file | Output: read the doc file | Pass |

For character recognition, another test case is designed i.e. test case 7 to test the character recognition process. The output of this test case is the recognized character corresponding to script. The results are shown in Figure 5.22. Hence, the test is passed. The last test case i.e.

test case 8 is used to read the recognized characters written in doc file. Figure 5.23 shows the results of this test case.


**Figure 5.16:** Line segmentation of Bilingual document from Academic domain

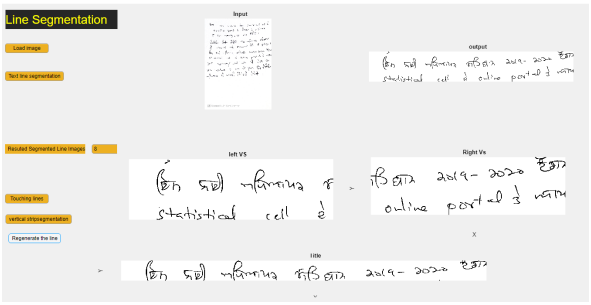



**Figure 5.17:** Line segmentation of skewed text in Bilingual document from Academic domain

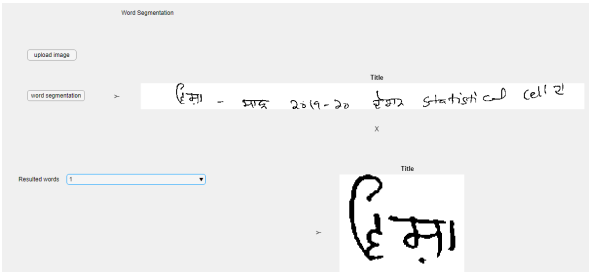



**Figure 5.18:** Word segmentation of Bilingual document from Academic domain

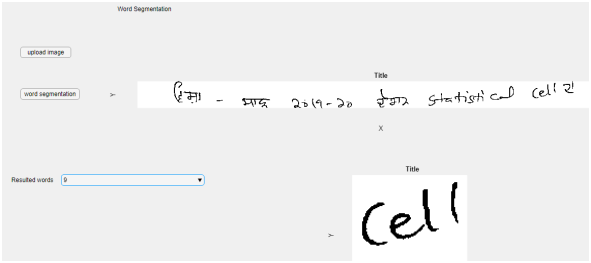



**Figure 5.19:** Word segmentation of Bilingual document from Academic domain

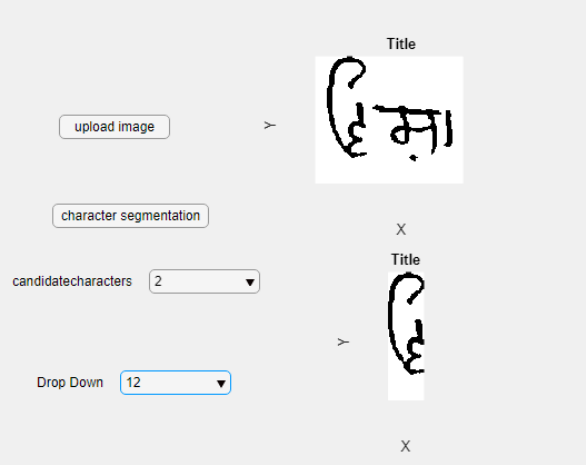



**Figure 5.20:** Character segmentation of Bilingual document from Academic domain

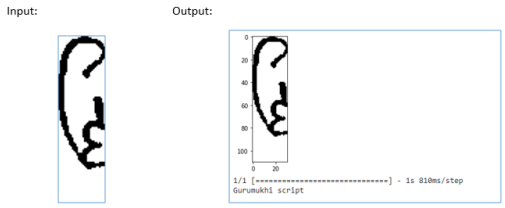



**Figure 5.21:** Script identification of segmented character of Bilingual document from Academic domain

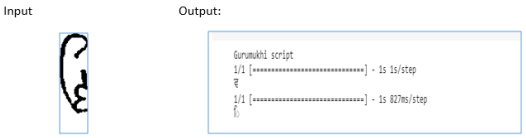



**Figure 5.22:** Character recognition of input data

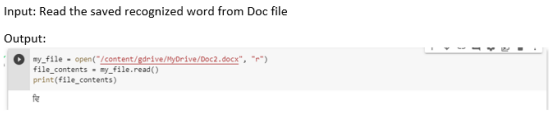



**Figure 5.23:** Read recognized characters

## 5.4.2 Feedback from potential users

The feedback from potential users has been collected based on these parameters: satisfaction, ease of use, accuracy, time, languages supported, efficiency of processing documents, and

some additional features or improvements. The users were satisfied with the performance of multilingual document recognition system, but the use of system was little complicated for them as some manual processing is needed. Further, they rated the accuracy good but time taken to process the documents is high and not satisfactory. For language support, the system is efficiently supporting all the required languages. Further for curved text, the system was not performing very efficiently. User manual should be there to guide the user regrading its working. Proper interface is needed where all the modules should be combined. The system should recognize the document directly through system i.e. either through PC camera or through mobile. User should be able to enter the feedback about its use for further improvements. The interface should provide immediate feedback to user actions. This could be through visual cues like changes in color, animation, or textual feedback, informing users about the outcome of their actions. It should reduce the manual work through refactoring of code, it should display messages for whatever errors are coming.

# Chapter 6

# Conclusions and Future scope

This chapter concludes the thesis and discusses the scope for future work.

## 6.1 Conclusion

The major objectives of this research work were to conduct the in depth review of the work done in the domain of monolingual and bilingual text recognition especially for Gurumukhi-English script and to propose the domain based bilingual handwritten text recognition system for the same. To achieve the first objective, an in depth literature review has been conducted for text recognition for Indic as well as non Indic scripts. Apart from this the review was conducted corresponding to each phase of text recognition like preprocessing, segmentation, feature extraction, classification and post processing for monolingual and multilingual text recognition. The unresolved issues at each phase has been identified and many of them have been resolved in the current work. Such as, the traditional methods use machine learning and statistical approaches to recognize the handwritten text, which are not efficient to recognize the curved, skewed and touching text with large variations in writing styles. Further, the use of latest approaches like deep learning is not a feasible option for regional languages due to the large training cost of deep networks and unavailability of diverse datasets for regional languages especially for Indic scripts etc.

The second objective was to propose a domain based bilingual handwritten text recognition system for Gurumukhi-English script including alphanumeric and special characters. In this

we have designed a text recognition system for Academic domain as it is the most common domain where bilingual documents are used. In the research work carried out so far, for bilingual text recognition of Gurumukhi-English script, printed text using traditional machine learning models has been widely considered. However, with the help of latest deep learning based approaches this work presents the domain based bilingual handwritten text recognition system for Gurumukhi-English scripts. The proposed work considers both the traditional as well as deep learning methods to design the system for bilingual text recognition of handwritten documents containing Gurumukhi-English script. For segmentation of handwritten documents simple computer vision approaches have been used without any trained machine learning models. It helps in reducing the overheads of training a model for new datasets.

The proposed system considers various sub processes of HTR to process the bilingual documents like segmentation, script identification and character recognition. Three segmentation algorithms have been proposed to segment the handwritten documents into lines, words and characters. All these algorithms have been designed considering the academic domain bilingual handwritten documents. Further, for script identification traditional and deep learning based models have been proposed. For character recognition, three OCRs have been designed corresponding to the scripts of text i.e., English, Gurumukhi and Alphanumeric. This work also proposes a new OCR for composite character recognition of Gurumukhi characters. The composite character dataset with 307 classes has been designed to recognize the composite characters. The third objective was to implement the proposed system in a usable form. For this we have designed our own Academic domain dataset of bilingual handwritten documents. This has been collected from the users of Academic domain with different age group, professional backgrounds. Further, this dataset can be enhanced to design a benchmark dataset for bilingual handwritten text. It can also enhance the further research in the same domain.

The fourth objective was to test and validate the proposed system. The proposed system is able to segment the documents with closed, skewed and touching text with an accuracy of 92.95% for line and 92.25% for word segmentation on bilingual handwritten document dataset as shown in chapter 5 Table 5.1, 5.2, 5.3. For IAM dataset, the proposed methods

have achieved an accuracy of 92.95% and 89% for line and word respectively. The script identification accuracy of 98.61% has been achieved using VGG19 features and bagging-based ensemble classifier as represented in Table 5.9-5.13. Similarly, for composite character recognition, accuracy of 98% and 99% for level 1 and level 2 respectively has been achieved using ResNet50 based features and SVM classifier as represented in table 5.15 and 5.16.

A number of other performance measures like precision, recall, F1 score, detection rate, recognition accuracy etc. have been used apart from accuracy. The proposed system has been trained on the bilingual handwritten documents from Academic domain. All the proposed algorithms and models have been trained on the bilingual handwritten document dataset as well as benchmark dataset like IAM.

Further, to test the functionality of proposed system, a case study for academic domain document text recognition has been performed in section 5.2 of chapter 5. A number of test cases have been designed to test the performance of proposed system. The results of test cases have proved the efficiency of proposed system.

The error analysis has also been conducted where we have seen the cases, the proposed model is not working efficiently. Moreover, the generalization of deep learning models is still an unresolved problem. As the cost of training the deep learning model is very high this will effect the applicability of the proposed model. In this research the performance of system is effected greatly with the quality of camera and paper as we are not using any preprocessing approaches here. So the results of segmentation phase will effect the results of further processes. As of now, we have implemented a simple GUI, to increase the usage of the system we need to implement it in the form of a online portal where people can use it. The latest trends like unsupervised learning, generative adversarial networks, or advancements in neural network architectures can greatly impact the area of text recognition and processing. For handwriting recognition, the most expensive thing is to find the labeled data with huge variations. To resolve this issue we can use unsupervised learning for feature extraction and representation learning which involves training the system on large amounts of unlabeled data to automatically discover meaningful patterns and representations in the handwriting. Further to enhance the accuracy of recognition, the power of generative AI

could be explored such as transformers can be employed for sequence modeling and classification. Further, attention mechanisms can be used to focus on relevant parts of the input sequence, making it more effective at recognizing long sequences of handwritten text. GANs can be employed to generate synthetic handwritten text samples, which can be used to augment the training data for the recognition system. By training the system on a combination of real and synthetic data, it can become more robust to variations in handwriting styles and improve its generalization capabilities. Additionally, GANs can be used for data augmentation to increase the diversity of the training dataset, which can help prevent overfitting and improve the system's performance on unseen data.

## 6.2 Future Work

Research is iterative and continuous procedure. The work presented in this thesis focuses on solving the problem of bilingual text recognition of handwritten documents for Indic scripts using deep learning and machine learning models. There are several directions in which this research work could be expanded. Some of the suggestions for the future work are as follows:

1. This research work considers only handwritten documents for text recognition. In future hybrid documents containing textual as well as non textual data can also be considered for recognition. These hybrid documents consist of mixed data having printed as well as handwritten text, multi column text and graphical data.

2. Deep learning based models like generative adversarial networks are highly used these days for text generation. These models can be used in future to generate the dataset for regional languages as well as segmentation of documents.

3. This work does not use any kind of language models for post processing of text recognition. These language models can be incorporated in future to improve the recognition results.

4. The findings of proposed method can be used easily into other domains beyond academic document analysis, such as legal document processing, historical text digitiza-

tion as our segmentation phase is independent to content, script and writing style. The change comes at classification phase where you need to retrain the models on other domains datasets. Further, the major change needs at post processing phase, if we are using generative AI powers then we need to use domain specific content.

5. Some other performance parameters like Levenshtein distance can be used for character recognition in future.

6. Further semantic tagging can be used in the restricted academia domain to aid the natural language processor to determine the context of the current word in the given sentence.

# Bibliography


[1] R. Plamondon and S. N. Srihari, "Online and off-line handwriting recognition: a comprehensive survey," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 1, pp. 63–84, 2000.

[2] N. Arica and F. T. Yarman-Vural, "An overview of character recognition focused on off-line handwriting," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 31, no. 2, pp. 216–233, 2001.

[3] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in online handwriting recognition," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 12, no. 8, pp. 787–808, 1990.

[4] C.-L. Liu, S. Jaeger, and M. Nakagawa, "'online recognition of chinese characters: the state-of-the-art," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 2, pp. 198–213, 2004.

[5] R. Jayadevan, S. R. Kolhe, P. M. Patil, and U. Pal, "Offline recognition of devanagari script: A survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 6, pp. 782–796, 2011.

[6] U. Pal, R. Jayadevan, and N. Sharma, "Handwriting recognition in indian regional scripts: a survey of offline techniques," *ACM Transactions on Asian Language Information Processing (TALIP)*, vol. 11, no. 1, pp. 1–35, 2012.

[7] D. Ghosh, T. Dube, and A. Shivaprasad, "Script recognition—a review," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 32, no. 12, pp. 2142–2161, 2010.

[8] K. Ubul, G. Tursun, A. Aysa, D. Impedovo, G. Pirlo, and T. Yibulayin, "Script identification of multi-script documents: a survey," *IEEE Access*, vol. 5, pp. 6546–6559, 2017.

[9] S. Jaeger and M. Nakagawa, "Two on-line japanese character databases in unipen format," in *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*. IEEE, 2001, pp. 566–570.

[10] A. Holzinger, C. Stocker, B. Peischl, and K.-M. Simonic, "On using entropy for enhancing handwriting preprocessing," *Entropy*, vol. 14, no. 11, pp. 2324–2350, 2012.

[11] V. Bansal and R. Sinha, "Segmentation of touching and fused devanagari characters," *Pattern recognition*, vol. 35, no. 4, pp. 875–893, 2002.

[12] T. Saba, A. Rehman, A. Altameem, and M. Uddin, "Annotated comparisons of proposed preprocessing techniques for script recognition," *Neural Computing and Applications*, vol. 25, no. 6, pp. 1337–1347, 2014.

[13] E. Kavallieratou, N. Fakotakis, and G. Kokkinakis, "New algorithms for skewing correction and slant removal on word-level [ocr]," in *ICECS'99. Proceedings of ICECS'99. 6th IEEE International Conference on Electronics, Circuits and Systems (Cat. No. 99EX357)*, vol. 2. IEEE, 1999, pp. 1159–1162.

[14] A. Vinciarelli and J. Luettin, "A new normalization technique for cursive handwritten words," *Pattern recognition letters*, vol. 22, no. 9, pp. 1043–1050, 2001.

[15] U.-V. Marti and H. Bunke, "Using a statistical language model to improve the performance of an hmm-based cursive handwriting recognition system," in *Hidden Markov models: applications in computer vision*. World Scientific, 2001, pp. 65–90.

[16] A. Rehman, D. Mohammad, G. Sulong, and T. Saba, "Simple and effective techniques for core-region detection and slant correction in offline script recognition," in *2009 IEEE International Conference on Signal and Image Processing Applications*. IEEE, 2009, pp. 15–20.

[17] R. M. Bozinovic and S. N. Srihari, "Off-line cursive script word recognition," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 11, no. 1, pp. 68–83, 1989.

[18] A. Elnagar and R. Alhajj, "Segmentation of connected handwritten numeral strings," *Pattern Recognition*, vol. 36, no. 3, pp. 625–634, 2003.

[19] R. G. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 18, no. 7, pp. 690–706, 1996.

[20] J. Hu, A. S. Rosenthal, and M. K. Brown, "Combining high-level features with sequential local features for on-line handwriting recognition," in *International Conference on Image Analysis and Processing*. Springer, 1997, pp. 647–654.

[21] P. Pudil, J. Novovičová, and J. Kittler, "Floating search methods in feature selection," *Pattern recognition letters*, vol. 15, no. 11, pp. 1119–1125, 1994.

[22] S. Dalal and L. Malik, "A survey of methods and strategies for feature extraction in handwritten script identification," in *2008 First International Conference on Emerging Trends in Engineering and Technology*. IEEE, 2008, pp. 1164–1169.

[23] R. M. Saabni and J. A. El-Sana, "Comprehensive synthetic arabic database for on/off-line script recognition research," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 16, no. 3, pp. 285–294, 2013.

[24] Y. Chherawala, P. P. Roy, and M. Cheriet, "Feature set evaluation for offline handwriting recognition systems: application to the recurrent neural network model," *IEEE transactions on cybernetics*, vol. 46, no. 12, pp. 2825–2836, 2015.

[25] A.-L. Bianne-Bernard, F. Menasri, R. A.-H. Mohamad, C. Mokbel, C. Kermorvant, and L. Likforman-Sulem, "Dynamic and contextual information in hmm modeling for handwritten word recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 10, pp. 2066–2080, 2011.

[26] R. A.-H. Mohamad, L. Likforman-Sulem, and C. Mokbel, "Combining slanted-frame classifiers for improved hmm-based arabic handwriting recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 7, pp. 1165–1177, 2008.

[27] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA transactions on Signal and Information Processing*, vol. 3, 2014.

[28] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[29] J. Angel Arul Jothi and V. Mary Anita Rajam, "A survey on automated cancer diagnosis from histopathology images," *Artificial Intelligence Review*, vol. 48, no. 1, pp. 31–81, 2017.

[30] J. Sternby, J. Morwing, J. Andersson, and C. Friberg, "On-line arabic handwriting recognition with templates," *Pattern recognition*, vol. 42, no. 12, pp. 3278–3286, 2009.

[31] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[32] A.-r. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE transactions on audio, speech, and language processing*, vol. 20, no. 1, pp. 14–22, 2011.

[33] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," *Information Fusion*, vol. 42, pp. 146–157, 2018.

[34] W. Yang, L. Jin, D. Tao, Z. Xie, and Z. Feng, "Dropsample: A new training method to enhance deep convolutional neural networks for large-scale unconstrained handwritten chinese character recognition," *Pattern Recognition*, vol. 58, pp. 190–203, 2016.

[35] I.-J. Kim and X. Xie, "Handwritten hangul recognition using deep convolutional neural networks," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 18, no. 1, pp. 1–13, 2015.

[36] S. Roy, N. Das, M. Kundu, and M. Nasipuri, "Handwritten isolated bangla compound character recognition: A new benchmark using a novel deep learning approach," *Pattern Recognition Letters*, vol. 90, pp. 15–21, 2017.

[37] B. Shi, X. Bai, and C. Yao, "Script identification in the wild via discriminative convolutional neural network," *Pattern Recognition*, vol. 52, pp. 448–458, 2016.

[38] S. B. Ahmed, S. Naz, M. I. Razzak, S. F. Rashid, M. Z. Afzal, and T. M. Breuel, "Evaluation of cursive and non-cursive scripts using recurrent neural networks," *Neural Computing and Applications*, vol. 27, no. 3, pp. 603–613, 2016.

[39] A. K. Bhunia, A. Konwer, A. K. Bhunia, A. Bhowmick, P. P. Roy, and U. Pal, "Script identification in natural scene image and video frames using an attention based convolutional-lstm network," *Pattern Recognition*, vol. 85, pp. 172–184, 2019.

[40] K. Kukich, "Techniques for automatically correcting words in text," *Acm Computing Surveys (CSUR)*, vol. 24, no. 4, pp. 377–439, 1992.

[41] J. F. Pitrelli and M. P. Perrone, "Confidence modeling for verification post-processing for handwriting recognition," in *Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition*. IEEE, 2002, pp. 30–35.

[42] Y.-X. Li, C. L. Tan, and X. Ding, "A hybrid post-processing system for offline handwritten chinese script recognition," *Pattern analysis and applications*, vol. 8, no. 3, pp. 272–286, 2005.

[43] C. Oprean, L. Likforman-Sulem, A. Popescu, and C. Mokbel, "Handwritten word recognition using web resources and recurrent neural networks," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 18, no. 4, pp. 287–301, 2015.

[44] F. Zamora-Martinez, V. Frinken, S. España-Boquera, M. J. Castro-Bleda, A. Fischer, and H. Bunke, "Neural network language models for off-line handwriting recognition," *Pattern Recognition*, vol. 47, no. 4, pp. 1642–1652, 2014.

[45] G. S. Lehal, "A bilingual gurmukhi-english ocr based on multiple script identifiers and language models," in *Proceedings of the 4th International Workshop on Multilingual OCR*, 2013, pp. 1–5.

[46] R. Pramanik and S. Bag, "Handwritten bangla city name word recognition using cnn-based transfer learning and fcn," *Neural Computing and Applications*, vol. 33, no. 15, pp. 9329–9341, 2021.

[47] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[48] M. Agrawal, K. Bali, S. Madhvanath, and L. Vuurpijl, "Upx: A new xml representation for annotated datasets of online handwriting data," in *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*. IEEE, 2005, pp. 1161–1165.

[49] D. Llorens, F. Prat, A. Marzal, J. M. Vilar, M. J. Castro, J.-C. Amengual, S. Barrachina, A. Castellanos, S. E. Boquera, J. Gómez *et al.*, "The ujipenchars database: a pen-based database of isolated handwritten characters." in *LREC*, 2008.

[50] C. Viard-Gaudin, P. M. Lallican, S. Knerr, and P. Binter, "The ireste on/off (ironoff) dual handwriting database," in *Document Analysis and Recognition, 1999. ICDAR'99. Proceedings of the Fifth International Conference on*. IEEE, 1999, pp. 455–458.

[51] J. J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 16, no. 5, pp. 550–554, 1994.

[52] U.-V. Marti and H. Bunke, "The iam-database: an english sentence database for offline handwriting recognition," *International Journal on Document Analysis and Recognition*, vol. 5, no. 1, pp. 39–46, 2002.

[53] A. Shivram, C. Ramaiah, S. Setlur, and V. Govindaraju, "Ibm_ub_1: a dual mode unconstrained english handwriting dataset," in *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*. IEEE, 2013, pp. 13–17.

[54] C.-L. Liu, F. Yin, D.-H. Wang, and Q.-F. Wang, "Casia online and offline chinese handwriting databases," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*. IEEE, 2011, pp. 37–41.

[55] Y. Li, L. Jin, X. Zhu, and T. Long, "Scut-couch2008: A comprehensive online unconstrained chinese handwriting dataset," *ICFHR*, vol. 2008, pp. 165–170, 2008.

[56] M. Kherallah, N. Tagougui, A. M. Alimi, H. El Abed, and V. Margner, "Online arabic handwriting recognition competition," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*. IEEE, 2011, pp. 1454–1458.

[57] N. E. B. Amara, O. Mazhoud, N. Bouzrara, and N. Ellouze, "Arabase: A relational database for arabic ocr systems." *Int. Arab J. Inf. Technol.*, vol. 2, no. 4, pp. 259–266, 2005.

[58] M. Kherallah, A. Elbaati, H. Abed, and A. Alimi, "The on/off (lmca) dual arabic handwriting database," in *11th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2008.

[59] S. Al Maadeed, W. Ayouby, A. Hassaïne, and J. M. Aljaam, "Quwi: An arabic and english handwriting dataset for offline writer identification," in *Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on*. IEEE, 2012, pp. 746–751.

[60] R. Hussain, A. Raza, I. Siddiqi, K. Khurshid, and C. Djeddi, "A comprehensive survey of handwritten document benchmarks: structure, usage and evaluation," *EURASIP Journal on Image and Video Processing*, vol. 2015, no. 1, p. 46, 2015.

[61] B. Nethravathi, C. Archana, K. Shashikiran, A. G. Ramakrishnan, and V. Kumar, "Creation of a huge annotated database for tamil and kannada ohr," in *Frontiers in Handwriting Recognition (ICFHR), 2010 International Conference on*. IEEE, 2010, pp. 415–420.

[62] S. Singh, A. Sharma, and I. Chhabra, "A dominant points-based feature extraction approach to recognize online handwritten strokes," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 20, no. 1, pp. 37–58, 2017.

[63] S. M. Obaidullah, C. Halder, K. Santosh, N. Das, and K. Roy, "Phdindic_11: page-level handwritten document image dataset of 11 official indic scripts for script identification," *Multimedia Tools and Applications*, vol. 77, no. 2, pp. 1643–1678, 2018.

[64] R. Guha, M. Ghosh, P. K. Singh, R. Sarkar, and M. Nasipuri, "A hybrid swarm and gravitation-based feature selection algorithm for handwritten indic script classification problem," *Complex & Intelligent Systems*, vol. 7, no. 2, pp. 823–839, 2021.

[65] M. Ghosh, S. S. Roy, H. Mukherjee, S. M. Obaidullah, X.-Z. Gao, and K. Roy, "Movie title extraction and script separation using shallow convolution neural network," *IEEE Access*, vol. 9, pp. 125 184–125 201, 2021.

[66] S. A. Mahmoud, I. Ahmad, M. Alshayeb, W. G. Al-Khatib, M. T. Parvez, G. A. Fink, V. Märgner, and H. El Abed, "Khatt: Arabic offline handwritten text database," in *2012 International Conference on Frontiers in Handwriting Recognition*. IEEE, 2012, pp. 449–454.

[67] S. Strassel, "Linguistic resources for arabic handwriting recognition," in *Proceedings of the Second International Conference on Arabic Language Resources and Tools, Cairo, Egypt*, 2009.

[68] A. Mezghani, S. Kanoun, M. Khemakhem, and H. El Abed, "A database for arabic handwritten text image recognition and writer identification," in *2012 international conference on frontiers in handwriting recognition*. IEEE, 2012, pp. 399–402.

[69] M. Pechwitz, S. S. Maddouri, V. Märgner, N. Ellouze, H. Amiri *et al.*, "Ifn/enit-database of handwritten arabic words," in *Proc. of CIFED*, vol. 2. Citeseer, 2002, pp. 127–136.

[70] S. Bhowmik, S. Malakar, R. Sarkar, S. Basu, M. Kundu, and M. Nasipuri, "Off-line bangla handwritten word recognition: a holistic approach," *Neural Computing and Applications*, vol. 31, no. 10, pp. 5783–5798, 2019.

[71] E. Grosicki and H. El-Abed, "Icdar 2011-french handwriting recognition competition," in *2011 International Conference on Document Analysis and Recognition*. IEEE, 2011, pp. 1459–1463.

[72] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[73] M. Bonyani, S. Jahangard, and M. Daneshmand, "Persian handwritten digit, character and word recognition using deep learning," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 24, no. 1, pp. 133–143, 2021.

[74] N. Nayef, Y. Patel, M. Busta, P. N. Chowdhury, D. Karatzas, W. Khlif, J. Matas, U. Pal, J.-C. Burie, C.-l. Liu *et al.*, "Icdar2019 robust reading challenge on multi-lingual scene text detection and recognition—rrc-mlt-2019," in *2019 International conference on document analysis and recognition (ICDAR)*. IEEE, 2019, pp. 1582–1587.

[75] T. E. de Campos, B. R. Babu, and M. Varma, "Character recognition in natural images," in *Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal*, February 2009.

[76] C. Viard-Gaudin, P. M. Lallican, S. Knerr, and P. Binter, "The ireste on/off (ironoff) dual handwriting database," in *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR'99 (Cat. No. PR00318)*. IEEE, 1999, pp. 455–458.

[77] S. Dhivya and U. G. Devi, "Tamizhi: Historical tamil-brahmi script recognition using cnn and mobilenet," *ACM TRANSACTIONS ON ASIAN AND LOW-RESOURCE LANGUAGE INFORMATION PROCESSING*, vol. 20, no. 3, 2021.

[78] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet, "Unipen project of on-line data exchange and recognizer benchmarks," in *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3-Conference C: Signal Processing (Cat. No. 94CH3440-5)*, vol. 2. IEEE, 1994, pp. 29–33.

[79] S. M. Obaidullah, C. Halder, K. Santosh, N. Das, and K. Roy, "Phdindic_11: page-level handwritten document image dataset of 11 official indic scripts for script identification," *Multimedia Tools and Applications*, vol. 77, no. 2, pp. 1643–1678, 2018.

[80] T. Saba, A. S. Almazyad, and A. Rehman, "Online versus offline arabic script classification," *Neural Computing and Applications*, vol. 27, no. 7, pp. 1797–1804, 2016.

[81] T. Saba, A. Rehman, A. Altameem, and M. Uddin, "Annotated comparisons of proposed preprocessing techniques for script recognition," *Neural Computing and Applications*, vol. 25, no. 6, pp. 1337–1347, 2014.

[82] L. O'Gorman, "The document spectrum for page layout analysis," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 15, no. 11, pp. 1162–1173, 1993.

[83] J.-Y. Yoo, M.-K. Kim, S. Y. Ban, and Y.-B. Kwon, "Line removal and restoration of handwritten characters on the form documents," in *Proceedings of the fourth international conference on document analysis and recognition*, vol. 1. IEEE, 1997, pp. 128–131.

[84] B. Yu and A. K. Jain, "A generic system for form dropout," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 11, pp. 1127–1134, 1996.

[85] M. Blumenstein, C. K. Cheng, and X. Y. Liu, "New preprocessing techniques for handwritten word recognition," in *Proceedings of the second IASTED international conference on visualization, imaging and image processing (VIIP 2002), ACTA Press, Calgary*, 2002, pp. 480–484.

[86] M. Côté, E. Lecolinet, M. Cheriet, and C. Y. Suen, "Automatic reading of cursive scripts using a reading model and perceptual concepts," *International Journal on Document Analysis and Recognition*, vol. 1, no. 1, pp. 3–17, 1998.

[87] E. Kavallieratou, N. Fakotakis, and G. Kokkinakis, "A slant removal algorithm," *Pattern Recognition*, vol. 33, no. 7, pp. 1261–1262, 2000.

[88] M. E. Morita, J. Facon, F. Bortolozzi, S. J. Garnés, and R. Sabourin, "Mathematical morphology and weighted least squares to correct handwriting baseline skew," in *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR'99 (Cat. No. PR00318)*. IEEE, 1999, pp. 430–433.

[89] Y. Kessentini, T. Paquet, and A. B. Hamadou, "Off-line handwritten word recognition using multi-stream hidden markov models," *Pattern Recognition Letters*, vol. 31, no. 1, pp. 60–70, 2010.

[90] H. S. Beigi, K. Nathan, G. J. Clary, and J. Subrahmonia, "Size normalization in on-line unconstrained handwriting recognition," in *Proceedings of 1st International Conference on Image Processing*, vol. 1. IEEE, 1994, pp. 169–173.

[91] B. Q. Huang, Y. Zhang, and M. T. Kechadi, "Preprocessing techniques for online handwriting recognition," in *Seventh International Conference on Intelligent Systems Design and Applications (ISDA 2007)*. IEEE, 2007, pp. 793–800.

[92] R. K. Bawa and R. Rani, "A preprocessing technique for recognition of online handwritten gurmukhi numerals," in *International Conference on High Performance Architecture and Grid Computing*. Springer, 2011, pp. 275–281.

[93] M. A. Refaey, "Background ruled-lines detection and removal in full-colored handwritten image documents," *International Journal of Image and Graphics*, vol. 15, no. 02, p. 1540006, 2015.

[94] 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.

[95] A. Rehman, D. Mohamad, and G. Sulong, "Implicit vs explicit based script segmentation and recognition: a performance comparison on benchmark database," *Int. J. Open Problems Compt. Math*, vol. 2, no. 3, pp. 352–364, 2009.

[96] A. Rehman and T. Saba, "Neural networks for document image preprocessing: state of the art," *Artificial Intelligence Review*, vol. 42, no. 2, pp. 253–273, 2014.

[97] S. Eskenazi, P. Gomez-Krämer, and J.-M. Ogier, "A comprehensive survey of mostly textual document segmentation algorithms since 2008," *Pattern Recognition*, vol. 64, pp. 1–14, 2017.

[98] T. Saba, A. Rehman, and M. Elarbi-Boudihir, "Methods and strategies on off-line cursive touched characters segmentation: a directional review," *Artificial Intelligence Review*, vol. 42, no. 4, pp. 1047–1066, 2014.

[99] F. C. Ribas, L. Oliveira, A. Britto Jr, and R. Sabourin, "Handwritten digit segmentation: a comparative study," *international journal on document analysis and recognition (IJDAR)*, vol. 16, no. 2, pp. 127–137, 2013.

[100] I. Sanasam, P. Choudhary, and K. M. Singh, "Line and word segmentation of handwritten text document by mid-point detection and gap trailing," *Multimedia Tools and Applications*, vol. 79, no. 41, pp. 30 135–30 150, 2020.

[101] V. Vučković and B. Arizanović, "Efficient character segmentation approach for machine-typed documents," *Expert Systems with Applications*, vol. 80, pp. 210–231, 2017.

[102] R. Ptak, B. Żygadło, and O. Unold, "Projection-based text line segmentation with a variable threshold," *International Journal of Applied Mathematics and Computer Science*, vol. 27, no. 1, 2017.

[103] S. Basu, C. Chaudhuri, M. Kundu, M. Nasipuri, and D. K. Basu, "Text line extraction from multi-skewed handwritten documents," *Pattern Recognition*, vol. 40, no. 6, pp. 1825–1839, 2007.

[104] R. P. dos Santos, G. S. Clemente, T. I. Ren, and G. D. Cavalcanti, "Text line segmentation based on morphology and histogram projection," in *2009 10th International Conference on Document Analysis and Recognition*. IEEE, 2009, pp. 651–655.

[105] M. Jindal, R. Sharma, and G. Lehal, "Segmentation of horizontally overlapping lines in printed gurmukhi script," in *2006 International Conference on Advanced Computing and Communications*. IEEE, 2006, pp. 226–229.

[106] S. Susan and K. Rachna Devi, "Text area segmentation from document images by novel adaptive thresholding and template matching using texture cues," *Pattern Analysis and Applications*, vol. 23, no. 2, pp. 869–881, 2020.

[107] Y. Sun, T. S. Butler, A. Shafarenko, R. Adams, M. Loomes, and N. Davey, "Word segmentation of handwritten text using supervised classification techniques," *Applied Soft Computing*, vol. 7, no. 1, pp. 71–88, 2007.

[108] J. Ryu, H. I. Koo, and N. I. Cho, "Word segmentation method for handwritten documents based on structured learning," *IEEE Signal Processing Letters*, vol. 22, no. 8, pp. 1161–1165, 2015.

[109] G. Louloudis, B. Gatos, I. Pratikakis, and K. Halatsis, "A block-based hough transform mapping for text line detection in handwritten documents," in *Tenth International Workshop on Frontiers in Handwriting Recognition*. Suvisoft, 2006.

[110] G. Louloudis, B. Gatos, I. Pratikakis, and C. Halatsis, "Text line and word segmentation of handwritten documents," *Pattern recognition*, vol. 42, no. 12, pp. 3169–3183, 2009.

[111] Y. Li, Y. Zheng, D. Doermann, and S. Jaeger, "Script-independent text line segmentation in freestyle handwritten documents," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 8, pp. 1313–1329, 2008.

[112] D. V. Sharma and G. S. Lehal, "An iterative algorithm for segmentation of isolated handwritten words in gurmukhi script," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 2. IEEE, 2006, pp. 1022–1025.

[113] J. Jo, H. I. Koo, J. W. Soh, and N. I. Cho, "Handwritten text segmentation via end-to-end learning of convolutional neural networks," *Multimedia Tools and Applications*, vol. 79, no. 43, pp. 32 137–32 150, 2020.

[114] J. I. Olszewska, "Active contour based optical character recognition for automated scene understanding," *Neurocomputing*, vol. 161, pp. 65–71, 2015.

[115] M.-C. Jung, Y.-C. Shin, and S. N. Srihari, "Machine printed character segmentation method using side profiles," in *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028)*, vol. 6. IEEE, 1999, pp. 863–867.

[116] A. Rehman and T. Saba, "Performance analysis of character segmentation approach for cursive script recognition on benchmark database," *Digital Signal Processing*, vol. 21, no. 3, pp. 486–490, 2011.

[117] E. B. Lacerda and C. A. Mello, "Segmentation of connected handwritten digits using self-organizing maps," *Expert systems with applications*, vol. 40, no. 15, pp. 5867–5877, 2013.

[118] I. A. Humied, "Segmentation accuracy for offline arabic handwritten recognition based on bounding box algorithm," *Communications on Applied Electronics*, vol. 5, no. 9, pp. 21–30, 2016.

[119] G. A. Farulla, N. Murru, and R. Rossini, "A fuzzy approach to segment touching characters," *Expert Systems with Applications*, vol. 88, pp. 1–13, 2017.

[120] B. Su and S. Lu, "Accurate recognition of words in scenes without character segmentation using recurrent neural network," *Pattern Recognition*, vol. 63, pp. 397–405, 2017.

[121] T. K. Bhowmik, S. K. Parui, U. Roy, and L. Schomaker, "Bangla handwritten character segmentation using structural features: a supervised and bootstrapping approach," *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, vol. 15, no. 4, pp. 1–26, 2016.

[122] X. Wei, S. Ma, and Y. Jin, "Segmentation of connected chinese characters based on genetic algorithm," in *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*. IEEE, 2005, pp. 645–649.

[123] T. Yamaguchi, S. Tsuruoka, T. Yoshikawa, T. Shinogi, E. Makimoto, H. Ogata, and M. Shridhar, "A segmentation system for touching handwritten japanese characters," in *Proceedings eighth international workshop on frontiers in handwriting recognition*. IEEE, 2002, pp. 407–412.

[124] A. Ul-Hasan, M. Z. Afzal, F. Shafait, M. Liwicki, and T. M. Breuel, "A sequence learning approach for multiple script identification," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*. IEEE, 2015, pp. 1046–1050.

[125] S. Kavitha, P. Shivakumara, G. H. Kumar, and C. Tan, "A robust script identification system for historical indian document images," *Malaysian Journal of Computer Science*, vol. 28, no. 4, pp. 283–300, 2015.

[126] A. Kacem and A. Saïdani, "A texture-based approach for word script and nature identification," *Pattern Analysis and Applications*, vol. 20, no. 4, pp. 1157–1167, 2017.

[127] D. Brodić, A. Amelio, and Z. N. Milivojević, "Identification of fraktur and latin scripts in german historical documents using image texture analysis," *Applied Artificial Intelligence*, vol. 30, no. 5, pp. 379–395, 2016.

[128] X.-k. Han, A. Aysa, H. Mamt, and K. Ubul, "Script identification of central asian printed document images based on nonsubsampled contourlet transform." *Engineering Letters*, vol. 25, no. 4, 2017.

[129] S. Prasad, G. K. Verma, B. K. Singh, and P. Kumar, "Basic handwritten character recognition from multi-lingual image dataset using multi-resolution and multi-directional transform," *International Journal of Wavelets, Multiresolution and Information Processing*, vol. 10, no. 05, p. 1250046, 2012.

[130] A. Aggarwal, K. Singh, and K. Singh, "Use of gradient technique for extracting features from handwritten gurmukhi characters and numerals," *Procedia Computer Science*, vol. 46, pp. 1716–1723, 2015.

[131] A. Jamil, A. Batool, Z. Malik, A. Mirza, and I. Siddiqi, "Multilingual artificial text extraction and script identification from video images," *Int. J. Adv. Comput. Sci. Appl*, vol. 7, no. 4, pp. 529–539, 2016.

[132] P. K. Singh, R. Sarkar, N. Das, S. Basu, M. Kundu, and M. Nasipuri, "Benchmark databases of handwritten bangla-roman and devanagari-roman mixed-script document images," *Multimedia Tools and Applications*, vol. 77, no. 7, pp. 8441–8473, 2018.

[133] R. Rani, R. Dhir, and G. S. Lehal, "Script identification of pre-segmented multi-font characters and digits," in *2013 12th international conference on document analysis and recognition*. IEEE, 2013, pp. 1150–1154.

[134] M. Kumar and S. R. Jindal, "A study on recognition of pre-segmented handwritten multi-lingual characters," *Archives of Computational Methods in Engineering*, vol. 27, no. 2, pp. 577–589, 2020.

[135] A. Pandey, S. Singh, R. Kumar, and A. Tiwari, "Handwritten script recognition using soft computing," *International Journal of Advanced Computer Science Electronics Engineering*, vol. 1, no. 6, pp. 6–11, 2012.

[136] B. Verma, M. Blumenstein, and M. Ghosh, "A novel approach for structural feature extraction: contour vs. direction," *Pattern Recognition Letters*, vol. 25, no. 9, pp. 975–988, 2004.

[137] M. Kumar, R. Sharma, and M. Jindal, "Efficient feature extraction techniques for offline handwritten gurmukhi character recognition," *National Academy Science Letters*, vol. 37, no. 4, pp. 381–391, 2014.

[138] P. Sahare and S. B. Dhok, "Multilingual character segmentation and recognition schemes for indian document images," *IEEE access*, vol. 6, pp. 10 603–10 617, 2018.

[139] R. Mandal, P. P. Roy, U. Pal, and M. Blumenstein, "Multi-lingual date field extraction for automatic document retrieval by machine," *Information Sciences*, vol. 314, pp. 277–292, 2015.

[140] N. Das, J. M. Reddy, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu, "A statistical–topological feature combination for recognition of handwritten numerals," *Applied Soft Computing*, vol. 12, no. 8, pp. 2486–2495, 2012.

[141] D. Khanduja, N. Nain, and S. Panwar, "A hybrid feature extraction algorithm for devanagari script," *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, vol. 15, no. 1, pp. 1–10, 2015.

[142] P. Shivakumara, Z. Yuan, D. Zhao, T. Lu, and C. L. Tan, "New gradient-spatial-structural features for video script identification," *Computer Vision and Image Understanding*, vol. 130, pp. 35–53, 2015.

[143] R. Kumar, R. K. Sharma, and A. Sharma, "Recognition of multi-stroke based online handwritten gurmukhi aksharas," *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences*, vol. 85, no. 1, pp. 159–168, 2015.

[144] J. Du, J.-F. Zhai, and J.-S. Hu, "Writer adaptation via deeply learned features for online chinese handwriting recognition," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 20, no. 1, pp. 69–78, 2017.

[145] R. Sarkhel, N. Das, A. Das, M. Kundu, and M. Nasipuri, "A multi-scale deep quad tree based feature extraction method for the recognition of isolated handwritten characters of popular indic scripts," *Pattern Recognition*, vol. 71, pp. 78–93, 2017.

[146] M. K. Sharma and V. P. Dhaka, "Pixel plot and trace based segmentation method for bilingual handwritten scripts using feedforward neural network," *Neural Computing and Applications*, vol. 27, no. 7, pp. 1817–1829, 2016.

[147] S. Vajda, K. Roy, U. Pal, B. B. Chaudhuri, and A. Belaid, "Automation of indian postal documents written in bangla and english," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 08, pp. 1599–1632, 2009.

[148] S. Tian, U. Bhattacharya, S. Lu, B. Su, Q. Wang, X. Wei, Y. Lu, and C. L. Tan, "Multilingual scene character recognition with co-occurrence of histogram of oriented gradients," *Pattern Recognition*, vol. 51, pp. 125–134, 2016.

[149] G. X. Tan, C. Viard-Gaudin, and A. C. Kot, "Information retrieval model for online handwritten script identification," in *2009 10th International Conference on Document Analysis and Recognition*. IEEE, 2009, pp. 336–340.

[150] Y. M. Alginahi, M. Mudassar, and M. N. Kabir, "An arabic script recognition system," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 9, no. 9, pp. 3701–3720, 2015.

[151] A. M. Elgammal and M. A. Ismail, "Techniques for language identification for hybrid arabic-english document images," in *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*. IEEE, 2001, pp. 1100–1104.

[152] B. Shi, X. Bai, and C. Yao, "Script identification in the wild via discriminative convolutional neural network," *Pattern Recognition*, vol. 52, pp. 448–458, 2016.

[153] N. R. Soora and P. S. Deshpande, "A novel local skew correction and segmentation approach for printed multilingual indian documents," *Alexandria engineering journal*, vol. 57, no. 3, pp. 1609–1618, 2018.

[154] D. Keysers, T. Deselaers, H. A. Rowley, L.-L. Wang, and V. Carbune, "Multi-language online handwriting recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1180–1194, 2017.

[155] W. Pan, C. Y. Suen, and T. D. Bui, "Script identification using steerable gabor filters," in *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*. IEEE, 2005, pp. 883–887.

[156] D. Chakraborty and U. Pal, "Baseline detection of multi-lingual unconstrained handwritten text lines," *Pattern Recognition Letters*, vol. 74, pp. 74–81, 2016.

[157] Y. Chherawala, P. P. Roy, and M. Cheriet, "Combination of context-dependent bidirectional long short-term memory classifiers for robust offline handwriting recognition," *Pattern Recognition Letters*, vol. 90, pp. 58–64, 2017.

[158] A. L. Spitz, "Determination of the script and language content of document images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 3, pp. 235–245, 1997.

[159] U. Pal and B. Chaudhuri, "Indian script character recognition: a survey," *pattern Recognition*, vol. 37, no. 9, pp. 1887–1899, 2004.

[160] D. Ghosh and A. Shivaprasad, "Handwritten script identification using possibilistic approach for cluster analysis," *Journal of the Indian Institute of Science*, vol. 80, no. 3, p. 215, 2000.

[161] K. Roy, U. Pal, and B. Chaudhuri, "Neural network based word-wise handwritten script identification system for indian postal automation," in *Proceedings of 2005 International Conference on Intelligent Sensing and Information Processing, 2005.* IEEE, 2005, pp. 240–245.

[162] K. Roy and K. Majumder, "Trilingual script separation of handwritten postal document," in *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*. IEEE, 2008, pp. 693–700.

[163] L. Zhou, Y. Lu, and C. L. Tan, "Bangla/english script identification based on analysis of connected component profiles," in *International Workshop on Document Analysis Systems*. Springer, 2006, pp. 243–254.

[164] K. Roy, S. K. Das, and S. M. Obaidullah, "Script identification from handwritten document," in *2011 Third National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics*. IEEE, 2011, pp. 66–69.

[165] K. Roy, A. Alaei, and U. Pal, "Word-wise handwritten persian and roman script identification," in *2010 12th International Conference on Frontiers in Handwriting Recognition*. IEEE, 2010, pp. 628–633.

[166] B. Dhandra and M. Hangarge, "Global and local features based handwritten text words and numerals script identification," in *International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007)*, vol. 2. IEEE, 2007, pp. 471–475.

[167] U. Pal, N. Sharma, T. Wakabayashi, and F. Kimura, "Handwritten numeral recognition of six popular indian scripts," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2. IEEE, 2007, pp. 749–753.

[168] M. I. Razzak, S. Hussain, and M. Sher, "Numeral recognition for urdu script in unconstrained environment," in *2009 International Conference on Emerging Technologies*. IEEE, 2009, pp. 44–47.

[169] A. Spitz and R. Furuta, "Multilingual document recognition electronic publishing, document manipulations, and typography," 1990.

[170] A. L. Spitz and M. Ozaki, "Palace: A multilingual document recognition system," in *Document Analysis Systems*, vol. 1. World Scientific Singapore, 1995, pp. 16–37.

[171] J. Ding, L. Lam, and C. Y. Suen, "Classification of oriental and european scripts by using characteristic features," in *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, vol. 2. IEEE, 1997, pp. 1023–1027.

[172] G. Peake and T. Tan, "Script and language identification from document images," in *Proceedings Workshop on Document Image Analysis (DIA'97)*. IEEE, 1997, pp. 10–17.

[173] B. Waked, S. Bergler, C. Y. Suen, and S. Khoury, "Skew detection, page segmentation, and script classification of printed document images," in *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)*, vol. 5. IEEE, 1998, pp. 4470–4475.

[174] J. Hochberg, P. Kelly, T. Thomas, and L. Kerns, "Automatic script identification from document images using cluster-based templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 176–181, 1997.

[175] S. Chaudhury and R. Sheth, "Trainable script identification strategies for indian languages," in *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR'99 (Cat. No. PR00318)*. IEEE, 1999, pp. 657–660.

[176] S. B. Patil and N. Subbareddy, "Neural network based system for script identification in indian documents," *Sadhana*, vol. 27, no. 1, pp. 83–97, 2002.

[177] Z. Chi, Q. Wang, and W.-C. Siu, "Hierarchical content classification and script determination for automatic document image processing," *Pattern Recognition*, vol. 36, no. 11, pp. 2483–2500, 2003.

[178] U. Pal and B. Chaudhuri, "Identification of different script lines from multi-script documents," *Image and Vision computing*, vol. 20, no. 13-14, pp. 945–954, 2002.

[179] S.-W. Lee and J.-S. Kim, "Multi-lingual, multi-font and multi-size large-set character recognition using self-organizing neural network," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 1. IEEE, 1995, pp. 28–33.

[180] V. Ablavsky and M. R. Stevens, "Automatic feature selection with applications to script identification of degraded documents." in *ICDAR*. Citeseer, 2003, pp. 750–754.

[181] C. Jawahar, M. P. Kumar, and S. R. Kiran, "A bilingual ocr for hindi-telugu documents and its applications," in *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.* IEEE, 2003, pp. 408–412.

[182] R. Kumar, V. Chaitanya, and C. Jawahar, "A novel approach to script separation," in *Proc. Int'l Conf. Advances in Pattern Recognition*, 2003, pp. 289–292.

[183] S. B. Moussa, A. Zahour, A. Benabdelhafid, and A. M. Alimi, "Fractal-based system for arabic/latin, printed/handwritten script identification," in *2008 19th International Conference on Pattern Recognition*. IEEE, 2008, pp. 1–4.

[184] S. Kanoun, A. Ennaji, Y. Lecourtier, and A. M. Alimi, "Script and nature differentiation for arabic and latin text images," in *Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition*. IEEE, 2002, pp. 309–313.

[185] M. Benjelil, S. Kanoun, R. Mullot, and A. M. Alimi, "Arabic and latin script identification in printed and handwritten types based on steerable pyramid features," in *2009 10th International Conference on Document Analysis and Recognition*. IEEE, 2009, pp. 591–595.

[186] A. Saïdani, A. K. Echi, and A. Belaid, "Identification of machine-printed and handwritten words in arabic and latin scripts," in *2013 12th International Conference on Document Analysis and Recognition*. IEEE, 2013, pp. 798–802.

[187] J. Gllavata and B. Freisleben, "Script recognition in images with complex backgrounds," in *Proceedings of the Fifth IEEE International Symposium on Signal Processing and Information Technology, 2005.* IEEE, 2005, pp. 589–594.

[188] D. Zhao, P. Shivakumara, S. Lu, and C. L. Tan, "New spatial-gradient-features for video script identification," in *2012 10th IAPR international workshop on document analysis systems*. IEEE, 2012, pp. 38–42.

[189] T. Q. Phan, P. Shivakumara, Z. Ding, S. Lu, and C. L. Tan, "Video script identification based on text lines," in *2011 International Conference on Document Analysis and Recognition*. IEEE, 2011, pp. 1240–1244.

[190] N. Sharma, R. Mandal, R. Sharma, P. P. Roy, U. Pal, and M. Blumenstein, "Multilingual text recognition from video frames," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2015, pp. 951–955.

[191] N. Sharma, U. Pal, and M. Blumenstein, "A study on word-level multi-script identification from video frames," in *2014 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2014, pp. 1827–1833.

[192] J. J. Lee, J. H. Kim, and M. Nakajima, "A hierarchical hmm network-based approach for on-line recognition of multi-lingual cursive handwritings," *IEICE TRANSACTIONS on Information and Systems*, vol. 81, no. 8, pp. 881–888, 1998.

[193] A. M. Namboodiri and A. K. Jain, "Online script recognition," in *2002 International Conference on Pattern Recognition*, vol. 3. IEEE, 2002, pp. 736–739.

[194] P. K. Singh, R. Sarkar, N. Das, S. Basu, M. Kundu, and M. Nasipuri, "Benchmark databases of handwritten bangla-roman and devanagari-roman mixed-script document images," *Multimedia Tools and Applications*, vol. 77, no. 7, pp. 8441–8473, 2018.

[195] A. Mukhopadhyay, P. K. Singh, R. Sarkar, and M. Nasipuri, "A study of different classifier combination approaches for handwritten indic script recognition," *Journal of Imaging*, vol. 4, no. 2, p. 39, 2018.

[196] S. Chanda, K. Franke, and U. Pal, "Identification of indic scripts on torn-documents," in *2011 International Conference on Document Analysis and Recognition*. IEEE, 2011, pp. 713–717.

[197] S. Chanda, O. R. Terrades, and U. Pal, "Svm based scheme for thai and english script identification," in *Ninth international conference on document analysis and recognition (ICDAR 2007)*, vol. 1. IEEE, 2007, pp. 551–555.

[198] P. B. Pati and A. Ramakrishnan, "Word level multi-script identification," *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1218–1229, 2008.

[199] S. Pal, A. Alireza, U. Pal, and M. Blumenstein, "Multi-script off-line signature identification," in *2012 12th International Conference on Hybrid Intelligent Systems (HIS)*. IEEE, 2012, pp. 236–240.

[200] A. Malaviya and L. Peters, "Fuzzy handwriting description language:: Fohdel," *Pattern Recognition*, vol. 33, no. 1, pp. 119–131, 2000.

[201] S. Thomas, C. Chatelain, L. Heutte, T. Paquet, and Y. Kessentini, "A deep hmm model for multiple keywords spotting in handwritten documents," *Pattern Analysis and Applications*, vol. 18, pp. 1003–1015, 2015.

[202] P. P. Roy, A. K. Bhunia, A. Das, P. Dey, and U. Pal, "Hmm-based indic handwritten word recognition using zone segmentation," *Pattern recognition*, vol. 60, pp. 1057–1075, 2016.

[203] K. Jayech, M. A. Mahjoub, and N. E. B. Amara, "Synchronous multi-stream hidden markov model for offline arabic handwriting recognition without explicit segmentation," *Neurocomputing*, vol. 214, pp. 958–971, 2016.

[204] A. Giménez, I. Khoury, J. Andrés-Ferrer, and A. Juan, "Handwriting word recognition using windowed bernoulli hmms," *Pattern Recognition Letters*, vol. 35, pp. 149–156, 2014.

[205] V. Ghods, E. Kabir, and F. Razzazi, "Fusion of hmm classifiers, based on x, y and (x, y) signals, for the recognition of online farsi handwriting: a large lexicon approach," *Arabian Journal for Science and Engineering*, vol. 39, no. 3, pp. 1713–1723, 2014.

[206] A. Sampath and N. Gomathi, "Fuzzy-based multi-kernel spherical support vector machine for effective handwritten character recognition," *Sādhanā*, vol. 42, no. 9, pp. 1513–1525, 2017.

[207] F. Simistira, V. Katsouros, and G. Carayannis, "Recognition of online handwritten mathematical formulas using probabilistic svms and stochastic context free grammars," *Pattern Recognition Letters*, vol. 53, pp. 85–92, 2015.

[208] H. Sajedi and M. Bahador, "Persian handwritten number recognition using adapted framing feature and support vector machines," *International Journal of Computational Intelligence and Applications*, vol. 15, no. 01, p. 1650004, 2016.

[209] X.-X. Niu and C. Y. Suen, "A novel hybrid cnn–svm classifier for recognizing handwritten digits," *Pattern Recognition*, vol. 45, no. 4, pp. 1318–1325, 2012.

[210] O. N. Al-Boeridi, S. Syed Ahmad, and S. Koh, "A scalable hybrid decision system (hds) for roman word recognition using ann svm: study case on malay word recognition," *Neural Computing and Applications*, vol. 26, no. 6, pp. 1505–1513, 2015.

[211] J. R. Prasad and U. Kulkarni, "Gujrati character recognition using weighted k-nn and mean $\chi$ 2 distance measure," *International Journal of Machine Learning and Cybernetics*, vol. 6, no. 1, pp. 69–82, 2015.

[212] P. Porwik, R. Doroz, and T. Orczyk, "The k-nn classifier and self-adaptive hotelling data reduction technique in handwritten signatures recognition," *Pattern Analysis and Applications*, vol. 18, no. 4, pp. 983–1001, 2015.

[213] Y. Yamashita and T. Wakahara, "Affine-transformation and 2d-projection invariant k-nn classification of handwritten characters via a new matching measure," *Pattern Recognition*, vol. 52, pp. 459–470, 2016.

[214] S. Bag, G. Harit, and P. Bhowmick, "Recognition of bangla compound characters using structural decomposition," *Pattern Recognition*, vol. 47, no. 3, pp. 1187–1201, 2014.

[215] X. Xiao, L. Jin, Y. Yang, W. Yang, J. Sun, and T. Chang, "Building fast and compact convolutional neural networks for offline handwritten chinese character recognition," *Pattern Recognition*, vol. 72, pp. 72–81, 2017.

[216] S. Zhang, L. Jin, and L. Lin, "Discovering similar chinese characters in online handwriting with deep convolutional neural networks," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 19, no. 3, pp. 237–252, 2016.

[217] S. Naz, A. I. Umar, R. Ahmad, S. B. Ahmed, S. H. Shirazi, I. Siddiqi, and M. I. Razzak, "Offline cursive urdu-nastaliq script recognition using multidimensional recurrent neural networks," *Neurocomputing*, vol. 177, pp. 228–241, 2016.

[218] S. Lai, L. Jin, and W. Yang, "Toward high-performance online hccr: A cnn approach with dropdistortion, path signature and spatial stochastic max-pooling," *Pattern Recognition Letters*, vol. 89, pp. 60–66, 2017.

[219] P. Singh, A. Verma, and N. S. Chaudhari, "Feature selection based classifier combination approach for handwritten devanagari numeral recognition," *Sadhana*, vol. 40, no. 6, pp. 1701–1714, 2015.

[220] P. K. Singh, R. Sarkar, and M. Nasipuri, "Correlation-based classifier combination in the field of pattern recognition," *Computational Intelligence*, vol. 34, no. 3, pp. 839–874, 2018.

[221] H. Salimi and D. Giveki, "Farsi/arabic handwritten digit recognition based on ensemble of svd classifiers and reliable multi-phase pso combination rule," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 16, no. 4, pp. 371–386, 2013.

[222] L. S. Oliveira, M. Morita, and R. Sabourin, "Feature selection for ensembles applied to handwriting recognition," *International Journal of Document Analysis and Recognition (IJDAR)*, vol. 8, no. 4, pp. 262–279, 2006.

[223] Y. Kessentini, S. BenAbderrahim, and C. Djeddi, "Evidential combination of svm classifiers for writer recognition," *Neurocomputing*, vol. 313, pp. 1–13, 2018.

[224] R. Elanwar, W. Qin, and M. Betke, "Making scanned arabic documents machine accessible using an ensemble of svm classifiers," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 21, no. 1, pp. 59–75, 2018.

[225] Y. Kessentini, T. Burger, and T. Paquet, "A dempster–shafer theory based combination of handwriting recognition systems with multiple rejection strategies," *Pattern Recognition*, vol. 48, no. 2, pp. 534–544, 2015.

[226] C. De Stefano, C. D'elia, A. S. Di Freca, and A. Marcelli, "Classifier combination by bayesian networks for handwriting recognition," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 05, pp. 887–905, 2009.

[227] R. Bertolami and H. Bunke, "Hidden markov model-based ensemble methods for offline handwritten text line recognition," *Pattern Recognition*, vol. 41, no. 11, pp. 3452–3460, 2008.

[228] S. Masoudnia, O. Mersa, B. N. Araabi, A.-H. Vahabie, M. A. Sadeghi, and M. N. Ahmadabadi, "Multi-representational learning for offline signature verification using multi-loss snapshot ensemble of cnns," *Expert Systems with Applications*, vol. 133, pp. 317–330, 2019.

[229] Z. Li, Z. Han, J. Xing, Q. Ye, X. Yu, and J. Jiao, "High performance person re-identification via a boosting ranking ensemble," *Pattern Recognition*, vol. 94, pp. 187–195, 2019.

[230] H. M. Balaha, H. A. Ali, M. Saraya, and M. Badawy, "A new arabic handwritten character recognition deep learning system (ahcr-dls)," *Neural Computing and Applications*, vol. 33, no. 11, pp. 6325–6367, 2021.

[231] Y.-H. Liu, C.-C. Lin, and F. Chang, "Language identification of character images using machine learning techniques," in *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*. IEEE, 2005, pp. 630–634.

[232] J. Hochberg, P. Kelly, T. Thomas, and L. Kerns, "Automatic script identification from document images using cluster-based templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 176–181, 1997.

[233] J. Hochberg, K. Bowers, M. Cannon, and P. Kelly, "Script and language identification for handwritten document images," *International Journal on Document Analysis and Recognition*, vol. 2, no. 2-3, pp. 45–52, 1999.

[234] S. Jaeger, H. Ma, and D. Doermann, "Identifying script on word-level with informational confidence," in *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*. IEEE, 2005, pp. 416–420.

[235] J. J. Lee and J. H. Kim, "A unified network-based approach for online recognition of multi-lingual cursive handwritings," in *Proc. Fifth Int'l Workshop Frontiers in Handwriting Recognition*, 1996, pp. 393–397.

[236] L. Shijian and C. L. Tan, "Script and language identification in noisy and degraded document images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 1, pp. 14–24, 2008.

[237] F. Zamora-Martinez, V. Frinken, S. España-Boquera, M. J. Castro-Bleda, A. Fischer, and H. Bunke, "Neural network language models for off-line handwriting recognition," *Pattern Recognition*, vol. 47, no. 4, pp. 1642–1652, 2014.

[238] S. Carbonnel and E. Anquetil, "Lexicon organization and string edit distance learning for lexical post-processing in handwriting recognition," in *Ninth International Workshop on Frontiers in Handwriting Recognition*. IEEE, 2004, pp. 462–467.

[239] F. Farooq, A. Bhardwaj, and V. Govindaraju, "Using topic models for ocr correction," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 12, no. 3, pp. 153–164, 2009.

[240] F. Wang, Q. Guo, J. Lei, and J. Zhang, "Convolutional recurrent neural networks with hidden markov model bootstrap for scene text recognition," *IET Computer Vision*, vol. 11, no. 6, pp. 497–504, 2017.

[241] S. Naz, A. I. Umar, R. Ahmad, S. B. Ahmed, S. H. Shirazi, and M. I. Razzak, "Urdu nasta'liq text recognition system based on multi-dimensional recurrent neural network and statistical features," *Neural computing and applications*, vol. 28, no. 2, pp. 219–231, 2017.

[242] M. H. Mahalat, A. F. Mollah, S. Basu, and M. Nasipuri, "Design of novel post-processing algorithms for handwritten arabic numerals classification," *International Journal of Applied Pattern Recognition*, vol. 4, no. 4, pp. 342–357, 2017.

[243] F. Biadsy, R. Saabni, and J. El-Sana, "Segmentation-free online arabic handwriting recognition," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 25, no. 07, pp. 1009–1033, 2011.

[244] A. Bharath and S. Madhvanath, "Hmm-based lexicon-driven and lexicon-free word recognition for online handwritten indic scripts," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 4, pp. 670–682, 2012.

[245] Q.-F. Wang, F. Yin, and C.-L. Liu, "Handwritten chinese text recognition by integrating multiple contexts," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 8, pp. 1469–1481, 2012.

[246] S. Sundaram and A. Ramakrishnan, "Bigram language models and reevaluation strategy for improved recognition of online handwritten tamil words," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 14, no. 2, p. 8, 2015.

[247] R. Xu, D. Yeung, W. Shu, and J. Liu, "A hybrid post-processing system for handwritten chinese character recognition," *International journal of pattern recognition and artificial intelligence*, vol. 16, no. 06, pp. 657–679, 2002.

[248] F. Farooq, A. Bhardwaj, and V. Govindaraju, "Using topic models for ocr correction," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 12, no. 3, pp. 153–164, 2009.

[249] I. Abdelaziz, S. Abdou, and H. Al-Barhamtoshy, "A large vocabulary system for arabic online handwriting recognition," *Pattern Analysis and Applications*, vol. 19, no. 4, pp. 1129–1141, 2016.

[250] D. Hládek, J. Staš, S. Ondáš, J. Juhár, and L. Kovács, "Learning string distance with smoothing for ocr spelling correction," *Multimedia Tools and Applications*, vol. 76, no. 22, pp. 24 549–24 567, 2017.

[251] R. Kumar and R. K. Sharma, "An efficient post processing algorithm for online handwriting gurmukhi character recognition using set theory," *International journal of pattern recognition and artificial intelligence*, vol. 27, no. 04, p. 1353002, 2013.

[252] M.-H. Lee, S.-H. Kim, G.-S. Lee, S.-H. Kim, and H.-J. Yang, "Correction for misrecognition of korean texts in signboard images using improved levenshtein metric," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 6, no. 2, pp. 722–733, 2012.

[253] R. Xu, D. S. Yeung, and D. Shi, "A hybrid post-processing system for offline handwritten chinese character recognition based on a statistical language model," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 19, no. 03, pp. 415–428, 2005.

[254] F. Wang, Q. Guo, J. Lei, and J. Zhang, "Convolutional recurrent neural networks with hidden markov model bootstrap for scene text recognition," *IET Computer Vision*, vol. 11, no. 6, pp. 497–504, 2017.

[255] S. Kundu, S. Paul, S. K. Bera, A. Abraham, and R. Sarkar, "Text-line extraction from handwritten document images using gan," *Expert Systems with Applications*, vol. 140, p. 112916, 2020.

[256] P. Jindal and B. Jindal, "Line and word segmentation of handwritten text documents written in gurmukhi script using mid point detection technique," in *2015 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS)*. IEEE, 2015, pp. 1–6.

[257] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 2921–2926.

[258] A. Aggarwal and C. Singh, "Zernike moments-based gurumukhi character recognition," *Applied Artificial Intelligence*, vol. 30, no. 5, pp. 429–444, 2016.