密级： 保密期限：

北京郵電大學

博士学位论文

题目：Deep Learning based Urdu Optical Character Recognition

学　　号：2013090004

姓　　名：Ibrar Ahmad (阿巴尔)

专　　业：计算机科学与技术

导　　师：Prof. Dr. Wang Xiaojie (王小捷)

学　　院：计算机学院

2017 年 04 月 18 日

中国・北京

密级： 保密期限：

北京郵電大學

博士学位论文

题目：Deep Learning based Urdu Optical Character Recognition

学 号：2013090004

姓 名：Ibrar Ahmad (阿巴尔)

专 业：计算机科学与技术

导 师：Prof. Dr. Wang Xiaojie (王小捷)

学 院：计算机学院

2017 年 04 月 18 日

Doctoral Dissertation of Beijing University of Posts and Telecommunications

Research on

Deep Learning based Urdu Optical Character Recognition

Submitted to Beijing University of Posts and Telecommunications

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

By

Ibrar Ahmad

Dissertation Supervisor: Prof. Wang Xiaojie

Major: Computer Science and Technology

Center for Intelligence of Science and Technology (CIST)

School of Computer, BUPT, China

March 2017

独创性（或创新性）声明

本人声明所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研 究成果，也不包含为获得北京邮电大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：Ibrar Ahmad Khan 日期：2017-09-22

关于论文使用授权的说明

本人完全了解并同意北京邮电大学有关保留、使用学位论文的规定，即：北京邮电大学拥有以下关于学位论文的无偿使用权，具体包括：学校有权保留并向国家有关部门或机构送交学位论文，有权允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，有权允许采用影印、缩印或其它复制手段保存、汇编学位论文，将学位论文的全部或部分内容编入有关数据库进行检索。（保密的学位论文在解密后遵守此规定）

本人签名：Ibrar Ahmad Khan 日期：2017-09-22

导师签名：[signature] 日期：2017-09-22

摘要

光学字符识别(Optical Character Recognition, OCR)是一种将手写或打印文本转换为机器编码文本的方法，在许多行业都有广泛的应用。随着模式识别、机器学习以及计算机视觉技术的发展，目前，包括汉语、西班牙语、英语、日语、俄语等在内的许多语言的 OCR 系统都具有较高的性能，达到了可以在在日常生活中进行实际使用的水平。然而,有一些由阿拉伯语字母书写的语言,如乌尔都语、波斯语、普什图语等，由于在形状和草体上的广泛变异性，给 OCR 带来了巨大的困难。虽然, 这些语言的 OCR 在过去的十年中取得了一些进展, 但是由于技术上存在的难题以及资源和研究人员的匮乏，其 OCR 的性能还不好，需要更多的研究人员进行更多的改进工作。

深层神经网络(Deep Neural Network, DNN)在过去的几年里在图像分类和识别任务中都取得了较好的性能，这主要归功于 DNN 具有的自动特征学习的能力。这些自动学习到的特性通常比人工设计的特征能更好地适应任务。同时，也使得人们可以不需要特征提取的专业知识或领域知识。目前，DNN 发展出了较多的模型，如自编码模型(Autoencoders)、堆栈式自编码模型(Stack Autoencoders)、卷积神经网络(Convolutional Neural Network, CNN),以及长短期记忆模型(Long Short-Term Memory, LSTM)等，这些模型具有不同的提取特征的能力，广泛用于不同的任务中。

本论文针对乌尔都语 OCR 中存在的问题，结合乌尔都语的特点采用深层神经网络技术进行建模，以提升乌尔都语 OCR 性能，取得的主要成果如下。

提出了一种新的线切割和连体字母切割的方法，该算法通过采用曲线分割和更好地分配点和附加符号的方式, 实现了更高的线切割和连体字母切割精度。

提出了一种基于去噪自编码器(Denoising Autoencoders)的连字字母特征提取方法来自动提取乌尔都语文本的连体字母书写特征。将其用于对连体字母的分类, 在输出层采用支持向量机(Support Vector Machine, SVM) 模型，在大规模数据集上的分类性能达到了 98%。

提出了一种基于门控双向长短期记忆(Gated Bi-direction Long Short-Memory,GBLSTM)网络的乌尔都语整句 OCR 模型，GBLSTM 将句子作为输入标签,在输出层采用 softmax，模型在 UPTI 数据集上取得 96% 的精确度,比其他流行的乌尔都语 OCT 模型具有更好的性能。

关键词: 乌尔都语,光学字符识别, 连体字母, 去噪自编码器, 门控双向长短期记忆。

DEEP LEARNING BASED URDU OPTICAL CHARACTER RECOGNITION

ABSTRACT

Due to unprecedented developments in machine learning coupled with pattern recognition and computer vision algorithms, very successful Optical Character Recognition (OCR) system can be seen in every part of daily life. Optical character recognition is a field of research in pattern recognition, artificial intelligence and computer vision. OCR used for the recognition of text documents is widely applicable in both research and industry. OCR is a way to convert typewritten, handwritten or printed text into machine-encoded text. OCR for many languages like Mandarin (Chinese), Spanish, English, Arabic, Japanese, Russian etc. are much more accurate and have numerous applications in daily life. However, there are some Arabic script languages like Urdu, Persian, Pashto, Balochi and Sindhi etc. that still need much more advancement and improvement in the OCR field. All these languages pose difficulties for researchers and developers in dealing with the wide variability of characters' shapes and cursiveness. Thus, it is an uphill task to devise and develop OCRs for such languages. Although, the research in this field has got some momentum from last decade but still the dilemma is the scarcity of resources and researchers.

Deep Neural Networks (DNN) is outperforming for classification and recognition tasks. This better performance is primarily due to automatic feature learning. These features often lead to better performance than human engineered feature. Also, there is no need expertise of features extraction, nor requires domain knowledge. Furthermore, features can be extracted from different domains with the help of same algorithm Autoencoders, stacked

autoencoders and Long Short Term Memory (LSTM), Bidirectional LSTM (BLSTM) are the forms of DNN incorporating multi-layered feature processing and learning.

The objective of this thesis is to improve Urdu OCRs by use state of the art machine learning techniques. Firstly, segmentation is improved by putting forward line and ligature segmentation algorithms. These algorithms performed with better accuracy by using thresholding method with a curved line split algorithm and better allocation of dots/diacritics. Secondly, recognition of Urdu text is performed on ligature and line levels by using deep learning methods. Autoencoders are employed for ligatures' feature extraction instead of human crafted features. For classification of ligature, softmax and SVM are employed at the output layers, achieved accuracy of 98%. LSTM networks are successfully employed for context based Urdu sentence recognition in existing contributions. Gated BLSTM (GBLSTM) networks is introduced and evaluated on UPTI datasets that yielded better results than other prevalent OCR techniques. Gated BLSTM takes sentences as input labelled with ligature and softmax at output layer recognized sentences with 96% accuracy. All prevalent context based sentence recognition contributions rely on character based LSTM.

KEYWORDS: Urdu text segmentation, Nastaleeq script segmentation, line and ligature segmentation, Urdu Nastaleeq ligature recognition, offline printed ligature recognition, Arabic script, Denoising Autoencoder, Deep learning Network, Classification, supervised learning, Urdu Nastaleeq sentence recognition, BLSTM, GBLSTM, Urdu OCR.

Biographical Sketch

Mr. Ibrar Ahmad received a Bachelor degree in Computer Science from University of Peshawar, Pakistan in 2002. He did his Master in Computer Science from Lahore University of Management sciences in December, 2005. He served as a lecturer at University of Peshawar and Hazara University, Pakistan from 2006 to 2010. He joined Department of Computer Science, University of Peshawar, Pakistan as an Assistant Professor from 2010 to 2013.

He pursued his doctoral degree under the supervision of Prof. Dr. Wang Xiao Jie from Center for Intelligence of Science and Technology (CIST), School of Computer Science and Technology, Beijing University of Posts and Telecommunications, Beijing, China. His research interests focus on Urdu text segmentation, Urdu Nastaleeq ligature recognition, Deep learning Network, Classification, supervised learning, and context learning. He will receive his Ph.D. degree in Computer Sciences in July 2017.

Table of Contents

List of Tables

List of Figures

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter 5

List of Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
BLSTM	Bidirectional Long Short Term Memory
BP	Back Propagation
BPN	Back Propagation Neural-network
BRNN	Bi-directional Recurrent Neural Network
CC	Chain Code
CENIP-UCCP	Center for Image Processing-Urdu Corpus Construction Project
CLE	Center for Language Engineering
CLS	Curved-Line-Split
CNN	Convolutional Neural Networks
CR	Character Recognition
CTC	Connectionist Temporal Classification
CV	Computer Vision
DA	Denoising Autoencoder
DCT	Discrete Cosine Transforms
DNN	Deep Neural Networks
FCC	Freeman Chain Codes
GBLSTM	Gated Bidirectional Long Short Term Memory
HMM	Hidden Markov model
HP	Horizontal Projection
ICA	Independent Component Analysis
KNN	K-Nearest Neighbor
LDA	Linear Discriminant Analysis
LSTM	Long Short Term Memory
MDLSTM	Multi-Dimensional Long Short Term Memory
ML	Machine Learning
MLP	Multi-Layer Perceptron

MN	Masking Noise
OCR	Optical Character Recognition
PCA	Principle Component Analysis
PR	Pattern Recognition
RBF	Radial Basis Function
ReLU	Rectified linear Unit
RNN	Recurrent neural network
SDA	Stacked Denoising Autoencoder
SIFT	Invariant Feature Extraction
SP	Salt-and-Pepper Noise
SVM	Support Vector Machine
ULR-SDA	Urdu Ligature Recognition Stacked Denoising Autoencoder
UNHD	Urdu Nastaleeq Handwritten Dataset
UPTI	Urdu Printed Text Image

CHAPTER 1

Optical Character Recognition

It is always been a great desire of human beings to make machines that mimic human behavior and assist them in performing tasks intelligently. One of such great invention is computer that generally can perform quantitative tasks better than human beings as opposed to the qualitative tasks. The field of Artificial Intelligence (AI) serves as the basis for performing human like qualitative tasks by computer. Human beings normally use sense of the sight more than any other sense in day to day life. Computer Vision (CV), Machine Learning (ML) and Pattern Recognition (PR) are areas of AI that make it possible for computer to mimic the sense of sight. One notable use of sense of sight is to recognize text. Human beings can recognize very efficiently text patterns, different scripts, fonts, variation in shape of characters and complex layouts of documents. Character Recognition (CR) is the branch of PR and uses techniques of CV and ML to convert the text images into machine readable form.

1.1 Background of Optical Character Recognition (OCR)

Retina scanner invented by Charles R. Carey in 1870 [1] that paved the way for Optical Character Recognition (OCR). John B. Flowers put forward a patent the "One-Eyed Machine Stenographer" [2] in 1916 that can read and write script. 1929 Gustav Tauschek created a device named "Reading Machine" [3] in 1929 by using a photo-sensor for getting template and matching it with already stored in its memory. Emanuel Goldberg's patent named "Statistical machine" [1] which converts the recognized characters into standard telegraph code. US Post Office mails are sorted by an OCR machine invented by Jacob Rabinow [4] in 1965. The first commercial omni-font OCR [5] created by Kurzweil Computer Products Inc. in 1974. By then, many companies and researchers have upheld this field with a great success. Modern OCR systems have used the modern hardware with state of the art pattern recognition and learning algorithms.

Document image analysis consists of algorithms and techniques that are applied to document images for obtaining computer understandable form. OCR software are form document image analysis used to recognize characters in a document image.

OCR is a research subfield of computer vision, artificial intelligence and pattern recognition [6]. Optical Character Recognition software performs converts document images of typed, printed, and handwritten text into computer readable form. It is extensively employed for collection of data from printed documents, receipts, invoices, bank statements, passports, visiting cards, mails, and many document images.

Generally, OCRs may be divided into two broad categories: online and offline. In offline character recognition system, first images of the text documents are created and stored in computer and then OCRs are applied for recognition. On the other side, online character recognition system, characters are being recognized at the time of creation. Offline or online OCR systems can further be used on handwritten character or optical characters resulting in Handwritten Character Recognition (HCR) and Optical Character Recognition (OCR), respectively.

The design of an OCR depends more upon the characteristics of underlying language's script. So, OCR systems can be developed for isolated or cursive script. Character do not join in languages using isolated script as opposed to cursive scripts where characters are joined to form words or sub words. It may be very difficult in some cursive script languages, like Nastaleeq Urdu script, to segment text up to character level because of context sensitivity, overlapping, merging of words/sub-words, non-equispacing among words, non-uniform gaps in between consecutive lines, and merging of text lines etc. Segmentation free recognition approaches are proposed for coping with the complexity of segmentation. Such approaches attempt to recognize the word or sub word as a whole without its segmentation into characters.

Different fonts for a same script may differ significantly. An OCR system that can recognize better for one specific font of a script is known as single font OCR system. As characteristics and shape of characters change with the change in font. There are systems that are aimed for multi font recognition, called as omni-font OCR systems. Omni font systems are trained by more generalized learning algorithms for better recognition of different fonts. General OCR block diagram is presented in Figure 1.1.

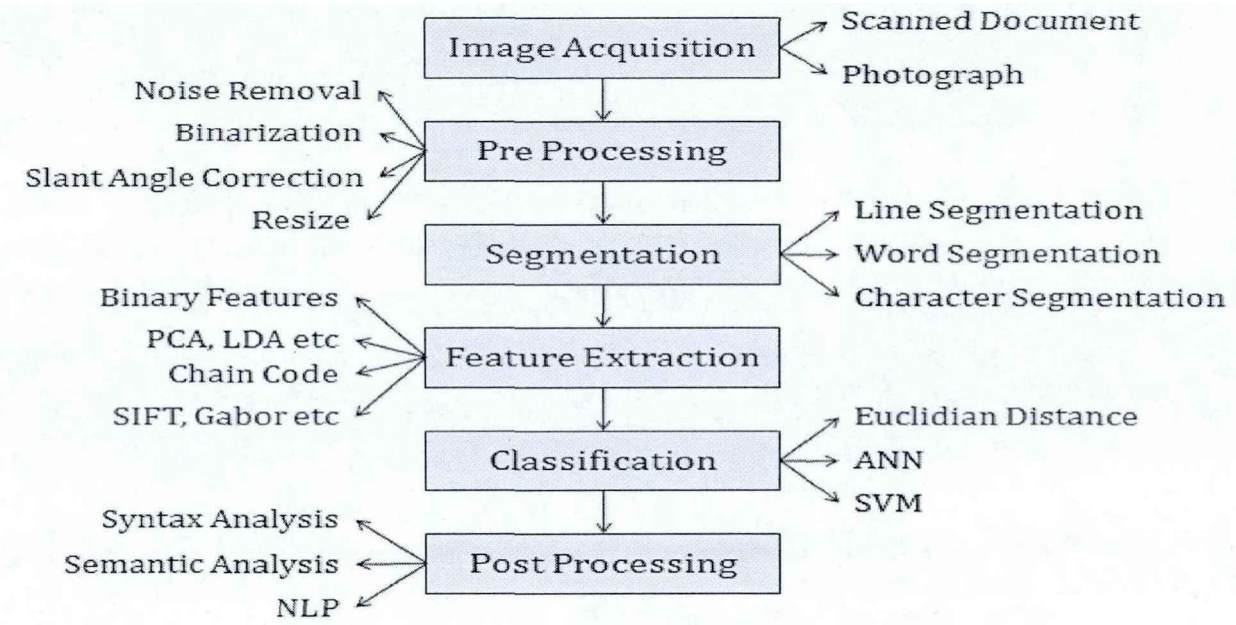


Fig 1.1. General Optical Character Recognition block diagram [7]

Brief explanation of the each step is as follows:

a) Image acquisition:

This step is also known as digitization. It is the process of acquiring images by specific hardware like scanners, cameras, or by using electronic pen. The source image may be of different forms such as printed, hand-written, typewritten document etc.

b) Preprocessing:

It is process of cleaning and enhancing images for subsequent step of segmentation and or feature extraction [3]. It may involve: the use of filters for noise removal, binarization for the acquiring resultant binary image, slant and skew correction, resizing while keeping aspect ratio. Skew is the image distortion by an angle or tilt i.e. the baseline angle of image is not perfectly horizontal. During the process of scanning, a page may tilt more towards one side. The resultant text in the scanned document image will also tilt to one side that needs to be corrected before applying recognition methodology of an OCR. So, detection and correction of skews is very crucial step for more successful results from an OCR system [8]. The process of binarization may be implemented easily with the help of thresholding mechanism.

In this method, a histogram of grey values of image is constructed with a cut-off point (a trough between the 2 crusts). All values of pixels measured more than the cut-off point are changed to 1 and rest all to zero [8].

c) Segmentation:

The recognition accuracy of OCR systems highly depends on the accuracy of segmentation. Generally, segmentation is performed for segmenting text and non-text parts, columns of text, text areas to lines, line to words/ligatures and ligature/ words to characters/strokes.

d) Feature Extraction:

Learning and classification depends upon better feature extraction. Better representation must represent every aspect of the image related to the learning task otherwise it can severely damage the learning and classification process. Feature can be extracted Principle Component Analysis (PCA), Independent Component Analysis (ICA), zoning, Linear Discriminant Analysis (LDA), Scale Invariant Feature Extraction (SIFT), Chain Code (CC), Gradient based techniques. Features for individual characters can be extracted by histogram method. Deep learning techniques help to extract better features than the hand engineered feature extraction methods.

e) Classification:

Features are fed to the classification algorithms for training and then it recognizes the class of the image examples included in test set. Such classifiers includes K-Nearest Neighbor (KNN), Support Vector Machine (SVM), and Artificial Neural Network (ANN) etc.

f) Post processing:

This is an optional step for enhancing the accuracy of recognition by checking results syntactically and semantically. N-gram method or any context based method can improve the text recognition.

1.2 Urdu Language

In last few decades, extensive work in the field of CR made possible the state-of-the-art OCR systems for many scripts like Latin, Greek, Mandarin (Chinese), and Japanese [6]. Some languages are given very less attention and still don't have accurate OCR system because of their complex writing styles. Such languages include Urdu, Persian, Arabic, Devanagari, Sanskrit, and Pashto etc.

Arabic scripts are used approximately by a quarter of world's population [9] in the form of many languages such as Arabic, Urdu, Persian, Punjabi, Sindhi, and Pashto etc. Urdu is the national language of Pakistan, and is widely spoken and understood in India and other Southern Asian countries as well. It is one of the top ten influential languages [10] of world. The number of research work is ongoing to preserve its literature. Urdu OCR, therefore, has great demand and receives more and more attentions.

1.2.1 Urdu Character Set

Urdu character set consists of 38 basic character. It does not include diacritics known as 'Aerabs' that are used guide pronunciation. Sometimes, Alif-Mad is also added to the character set. The alphabets of Urdu language are shown in the Figure1.2.

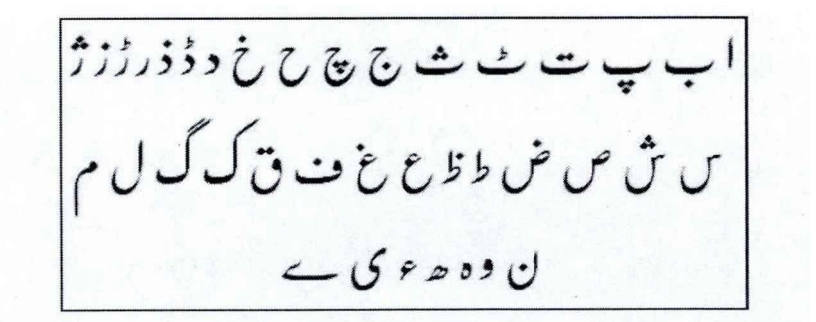


Fig. 1.2. Urdu Character Set

1.2.2 Diacritics

Urdu characters are also augmented with some special marks, called diacritics. The most important type of diacritics is dot ('Nuqta'). Dots are used to differentiate characters having basic similar shapes. Dots may be one to three in numbers and placed on a specific

position major stroke of a character. So, dots are intrinsic part of a character. Some characters don't have any dot at all.

There are diacritics representing vowel signs called "Aerabs" that are used to guide the pronunciation of characters in an Urdu word. In Figure 1.3, the circles portray characters to clarify the position of diacritic [11]. Aerabs may be Zabr, Zair, Paish, Khari Zabr, Juzm, Shud as read from right to left in Figure 1.3. 'Aerab' are mostly not used in Urdu text until and unless there are two or more pronunciations of the same word in same context.

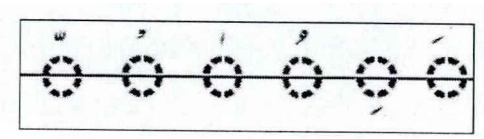


Figure 1.3 Aerabs for Urdu Language

1.2.3 Writing Styles

Urdu is an Arabic script language and can be written with different styles. Some of the writing styles used in Urdu calligraphy are depicted in Figure 1.4.

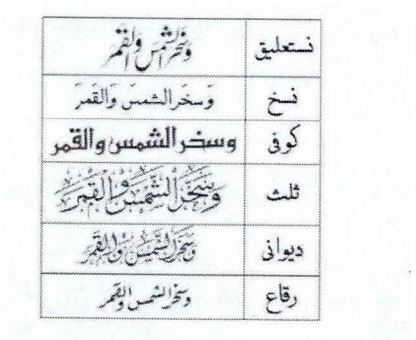


Figure 1.4 Writing styles for Urdu Language [12]

Two different writing styles are mainly in practice for Urdu language, namely Naskh and Nastaleeq. The former is written in a way that every character touches a horizontal baseline while in Nastaleeq characters in words are written diagonally directed from top right to bottom left corner [13]. Nastaleeq style of writing is generally used in traditional newspaper and books writings. Nastaleeq writing style is also followed in many other languages including Persian, Pashto, Panjabi, Baluchi, and Siraiki [13, 14]. Nastaleeq OCR, therefore, has great demand and receives more and more attentions.

1.3 Complexities of Nastaleeq Urdu Text

Nastaleeq accommodates more characters in less space because of its diagonality and cursive nature [14]. However, it is hard to segment Nastaleeq text into characters perfectly, as it is very cursive in nature. Moreover, characters recognition becomes more difficult because of its context sensitivity, compactness, overlapping, diagonality and several other characteristics [11, 15, 16] as shown in Fig. 1.5 and explained in detail subsequently.

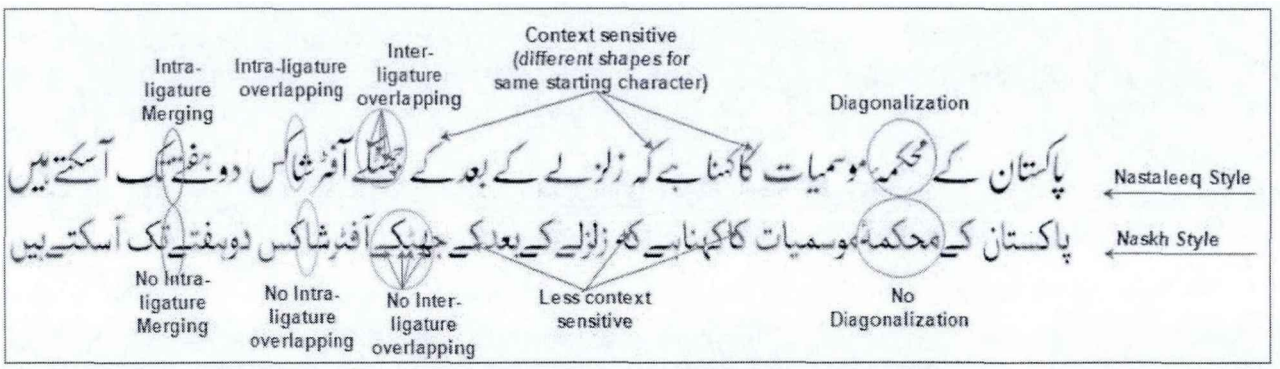


Fig. 1.5. Complexities of Nastaleeq writing style

The detailed discussion of the characteristics are given as follows.

a) Diagonality and Stacking

In Nastaleeq writing style, characters are stacked from right to left and top to bottom into a word. Characters are written diagonally into a word or subword. A subword that may contain connected (normally 1 to 8) characters is normally known as ligature in Urdu. A word in Urdu language consists of one or more ligatures. Width and height of a ligature may differ depending upon the number of characters and type of characters it contains. Consumption of less horizontal space is the main advantage of diagonality and stacking.

Fig. 1.6. Diagonality and stacking of characters [17]

The Figure 1.6 illustrates diagonality and stacking of characters in a word. A systematic way of formation of word is shown in seven steps. In every next step, one more character is added starting from one character. While adding a new character, the former

characters are stacked up towards the top right corner. A Nastaleeq Urdu sentence is shown in Figure 1.7, where arrows represent diagonality and stacking of characters.

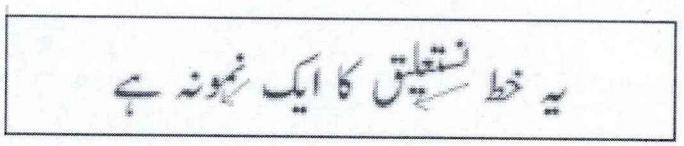


Fig. 1.7. A Nastaleeq Urdu sentence, Arrows represent diagonality and stacking of characters

b) Bi-directionality

Urdu script recognition is more complex because this is a bidirectional language. Words inside sentences are being written and read starting form right to left, whereas, numbers are written from left to right as depicted in Figure 1.8.

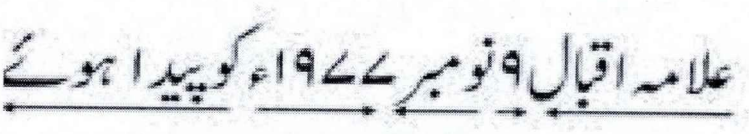
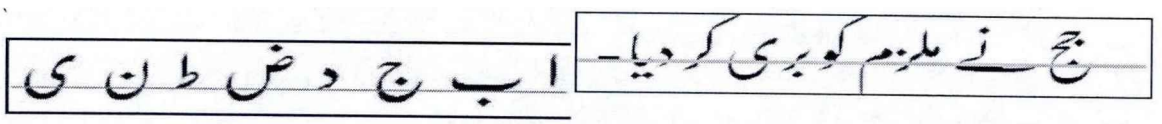


Fig. 1.8. Directionality of Nastaleeq Urdu sentence, Arrows represent directions of words and numbers

c) Hidden baseline

Nastaleeq writing style manages a hidden base line while at the same time adhering to the property of diagonality. The written text of Urdu flows on a hidden baseline. As ligatures of text start from top right and end at bottom left by maintaining a hidden baseline while keeping the text parallel to x-axis. Different characters are written at different vertical distances from this baseline. An example of isolated characters is shown in Figure 1.9(a) where each character has a specific starting and ending position in accordance with this baseline.



(a) (b)

Fig. 1.9 Hidden baseline for (a) characters and (b) Urdu sentence

As shown in the Figure 1.9, the hidden baseline dictates the correct placement of characters as well as ligatures for writing Urdu Nastaleeq. The position of isolated characters is fixed according to baseline. In case of ligature, only last character of a ligature takes its original position according to baseline and all other characters are stacked on this baseline character as depicted in Figure 1.9(b).

d) Joining and shape conversion

A ligature is made up of multiple characters, where shape of a character changes based upon its position inside a ligature. Urdu character set can be categorized into joiners and non-joiners characters as shown in Figure 1.10. Non-joiners are class of characters that can either come at the end of a ligature or can appear in isolation.

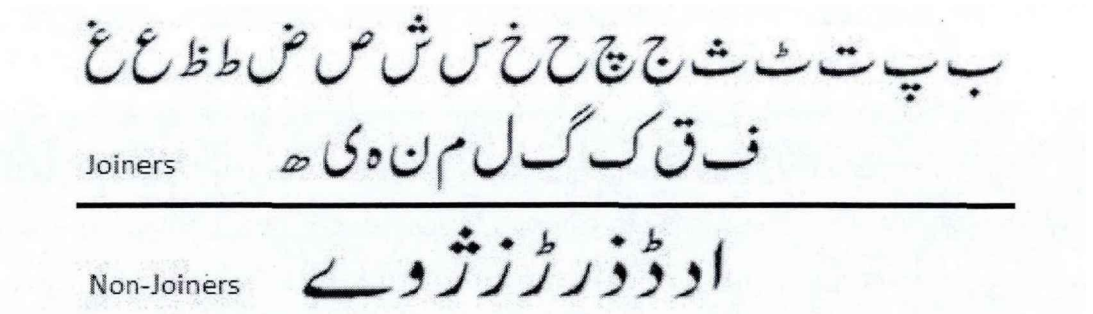


Figure 1.10 Joiners and non-joiners characters of Urdu Language

A character in a ligature may exhibit only 4 possible shapes in Naskh as shown 1.11. As, joiner characters of Naskh can be written in four different shapes namely initial, middle, final, and isolated [14].

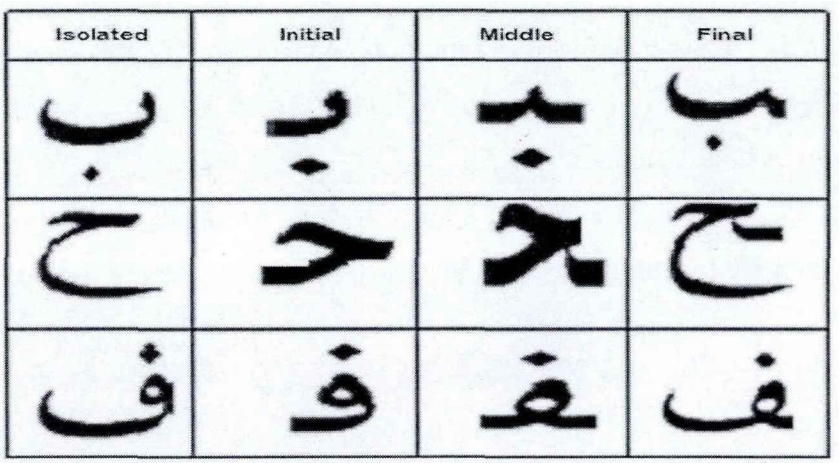


Fig. 1.11 Four different shapes (Isolated, initial, middle and final) of 3 (Naskh joiner) characters

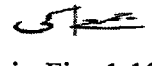
On the other hand, shapes of Urdu language (Nastaleeq) characters may approach to 50 [11]. For example, there are many different forms of initial, middle and final "bay" character as shown in Figure 1.12.

Class #		1	2	3	4	5	6	7	8	9	10	11
2 ب	Initial											
	Middle											
	Final											
Class #		12	13	14	15	16	17	18	19	20	21	
2 ب	Initial											
	Middle											
	Final											

Figure 1.12 Different shapes of "ب" appearing at initial middle and final positions [18]

The phenomenon of shape changing is dependent on addition of characters before and after a particular character. Generally, most of the characters coming at the end of a ligature adapt a shape similar to their isolated one as shown in Fig. 1.12. But, there is no hard and fast rule that characters at the end of ligatures are always in their original shape. Some of these characters acquire new shapes when joined.

e) Overlapping

Urdu Nastaleeq is always written diagonally by stacking preceding character of a ligature on succeeding in top to bottom manner [19] as shown in Fig 1.13. One of the property of Urdu Nastaleeq is overlapping of characters of ligatures and  words. The characters of two consecutive ligatures may overlap as shown in Fig. 1.13 and it is known as inter ligature overlapping. Similarly, there may be overlapping among the characters of a same ligature and it is known as inter-ligature overlapping as 'ب، ج' ' and 'ع' is overlapping 'ے ' in the second ligature(from right to left) of the Fig. 1.13.

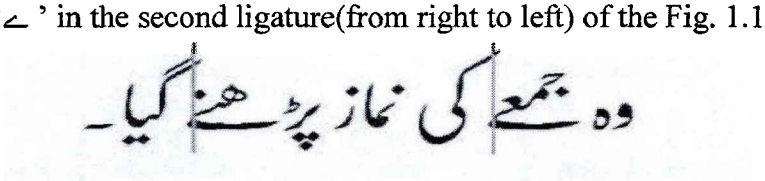


Fig 1.13 Inter and intra ligature overlapping

The placement of diacritics may change because of joining and overlapping of characters Diacritics and dots are differentiate among characters of same class. Same class of characters have same main stroke but containing different types diacritics and number of dots for differentiation. Figure 1.14 (a) portrays the shifting of three dots of 'چ' when it is joined with other characters. The major issue is experienced when two or more same characters are joined that contains dots as shown in Figure 1.14(b).

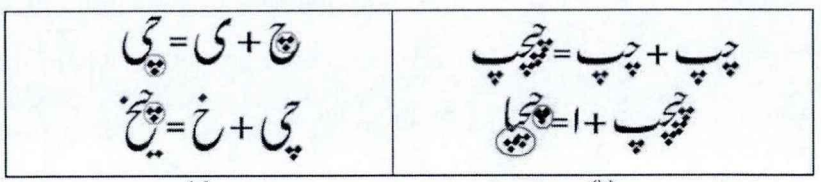
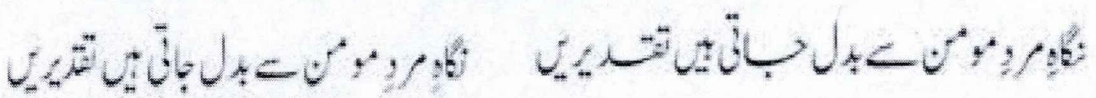


Fig 1.14 Effect of ligature overlapping on dots placement of characters [19]

f) Stretching

One of the unpredictable characteristics of Nastaleeq writing style is stretching. Standard Nastaleeq script do not use stretching, yet it is frequently used in the title and headlines of newspapers, books, magazines and art work. Stretching is the replacement of longer character versions instead of their standard [14]. Mostly characters only alter width upon stretching but some Urdu characters can even alter the default shape e.g. seen (س) as shown in Fig 1.15(a). However, it should be noted that not every character can be stretched for example (د) daal, (ر) ray, (ا) alif as depicted in Fig 1.15(b) and (c). The main aim of stretching is make the text eye catching and also to fill unnecessary space. Therefore, stretching increases the complications of character recognition tremendously. In Fig 1.15(b) and (c), stretched and unstarched versions of (ق) qaaf and (ج) jeem characters are shown.

Fig 1.15 (a) Stretched and non-stretched version of character seen [14]

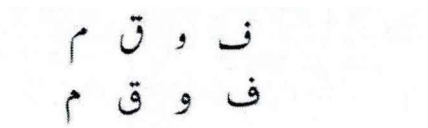


b) Non-Stretched version c) Stretched version

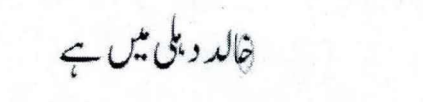
Fig 1.15 Stretched and non-stretched version of a sentence in Nastaleeq

g) Filled and false loops

There are some Urdu characters having loops in their primary stroke, main body of a character, such as Fey, Wao, Qaaf and Meem as shown in Fig 1.16(a). In Nastaleeq characters, such loops are filled while these are open in Naskh. This characteristic of Nastaleeq causes different characters of identical shape, e.g. dal and wow. The starting point of main stroke of some characters joins with the base line resulting in false loops, like Jeem, Chey, Hey, Khey, as shown in Fig 1.16(b). These properties add to the complexities of Urdu Nastaleeq text to be recognized by machine.



(a) Loops



(b) False loop

Fig 1.16 Filled and false loops [14]

h) Non-uniform spacing

In Urdu Nastaleeq script, there is non-uniform spacing between consecutive words or ligatures as shown in Fig 1.17. This is the main reason that words cannot be demarcated on the basis of the found spaces inside Nastaleeq text.

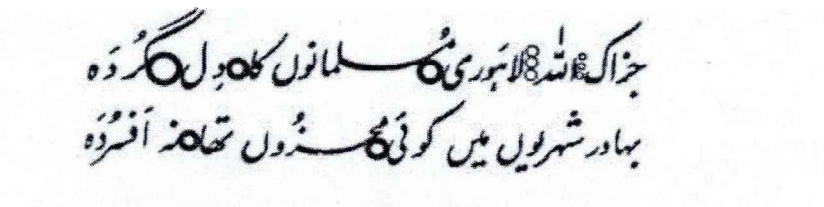


Fig 1.17 Non-uniform spacing [20]

Apart from aforementioned issues, there are many other complexities of Urdu Nastaleeq text, like direction of segmentation, segmentation cue point, multiple baselines,

and non-uniform interline spacing etc., which need to be resolved for Urdu Character Recognition.

1.4 Segmentation of Urdu Nastaleeq text

A document image is usually a collection of text and non-text parts. Text area contains headings, paragraph, and sentences. Each sentence is a collection of words composed of joined character in the form of ligatures and words. After correcting the skewed document image, next important step of layout analysis is segmentation. In this step, the identified text areas during layout analysis are segmented to paragraphs, sentences, words, ligatures and characters. Segmentation is a tough task for Urdu because of the complexities mentioned in section 1.3. The segmentation of Urdu Nastaleeq text images into character and ligatures is comparatively complex as compared to Naskh based languages. So, it is not straightforward to extend and use the works already done latter languages for Urdu language.

Accuracy of an OCR system depends upon the accurate segmentation method. Different guidelines are outlined in [54] for improvement of pre recognitions processes for Urdu Nastaleeq and claimed 100% and 94% accuracies for baseline identification and ligature identification, respectively. The horizontal profiling used for finding primary and secondary connected components and baseline [23]. Connected component method augmented with centroid-to-centroid approach [16] applied for component labeling.

A segmentation based approach [26] for Nastaleeq script recognition of 36 font size, segmented diacritics and main bodies before applying thinning. The main body of ligatures are segmented at branching point. Horizontal projection method [55, 56] is used for ligature extraction from Urdu Nastaleeq text images. Primary and secondary ligatures are separated by applying Freeman chain codes (FCC) with vertical scanning [57]. Seen and unseen words are segmented with the accuracies of 96.10% and 65.63%, respectively, by using trigram probabilities [58] by using a manually developed and cleaned corpus.

Detailed discussion about segmentation of Urdu Nastaleeq images is discussed in Chapter 3. All layout analysis with lines and ligature segmentation algorithms are discussed in comparison with the proposed line and ligature segmentation algorithms.

1.5 OCRs for Urdu Language

It is very uphill task to design strategies for Urdu Nastaleeq OCR in the presence of aforementioned complexities of this script. From last two decades researchers are taking keen interest in automatic recognition of printed Urdu script [13, 14, 21-24]. Center for Language Engineering (CLE), Pakistan, devoted for linguistic and computational development of Asian languages, especially Urdu language. Series of serious efforts can be seen in the form of dissertations [6, 9, 18, 19, 25-28] devoted for the uplift of Urdu OCR research, put forward by very competent researchers.

In prevalent contributions, the reported Urdu OCR methodologies can be characterized into analytical and holistic [29]. The former is also known as segmentation based and latter as segmentation-free.

Adnan Ul-Hasan [6] divided segmentation based OCR into two major classes namely: template matching and over-segmentation methods. In first class, characters are extracted and matched against the stored templates [33-36]. The recognition is based on a certain match criterion. The second segmentation based method is over-segmentation technique that depends upon finding an approximate demarcation point (cut) between two consecutive characters. The demarcation point between two consecutive characters is found by 3 methods [37] namely: (i) vertical projection (ii) feature-based boundary finding methods and (iii) skeleton based methods.

Naz et al. [29] divided analytical methodologies are into explicit and implicit segmentation techniques. Former techniques segment text up to character level and recognize underlying text based upon the recognition of characters. However, it is very hard to segment Nastaleeq text into characters accurately due to very cursive nature of Urdu Nastaleeq text [13, 14]. Moreover, characters recognition becomes more difficult because of its context sensitivity, compactness, overlapping, diagonality and several other characteristics [11, 15, 16] as detailed in preceding section. Therefore, accurate segmentation is crucial for better performance. Whereas, in implicit segmentation techniques, ground truths along with corresponding images(of text lines, words or ligatures) are given to machine learning algorithms. Implicit segmentation methods don't need segmentation of text. These methods identify segmentation cue points for labels inside given image at recognition time [30].

In Urdu script, there is no definite inter word spacing, therefore, it is hard to extract and subsequently learn words directly from text [14]. Urdu words are composed of one or more ligatures. Every ligature is separated from other ligatures with a space. So, recognition of words could be possible by learning ligatures followed by post processing that incorporates dictionary validation [29]. A ligature is composed of one or more characters that can easily be extracted from text as compared to characters. Holistic approaches or segmentation free methodologies mainly rely on recognition of ligature. In such methods, the shape of the ligature or sub-word is learned without segmenting it into characters. In segmentation free methodologies, each ligature is independently recognized by an OCR system. Such methodologies have the problem of scalability as system will unable to recognize newly added ligatures or sub-word to a language. Another issue with Urdu language is the huge number of candidate ligature classes i.e. approximately 26,000 ligatures [31]. Thus recognition of huge number of classes by segmentation free methodologies is the main concern for performance. However, such systems can perform better for applications where minimum number of ligature classes are needed like bank check and postal addresses recognition. Detailed discussion about aforementioned methodologies of Urdu OCRs are given as follows.

1.5.1 Segmentation based Approach

In segmentation-based OCR approach, the text is segmented into language unit, mostly individual characters, appropriate for training and subsequently for recognition. This approach depends upon accurate extraction of individual language characters. One of the popular open-source OCR system, Tesseract [32] that is built on segmentation-based approach. Naz et al. [29] divided segmentation based Urdu language OCR systems into explicit and implicit segmentations methods. Adnan Ul-Hasan [6] named the former as template matching and latter as over segmentation methods. For the sake of clarity, following sections introduce the segmentation based methods as explicit and implicit segmentation methods.

a) Explicit Segmentation based OCR

In explicit segmentation based approach, the underlying text to be recognized is segmented into characters/independent units/strokes. This approach is costly in terms of time consumption for performing segmentation. Segmentation of Urdu Nastaleeq is a

difficult task because of its complexities (see Section 1.3). General idea such methods is given by Adnan ul hassan [6] in Fig 1.18.

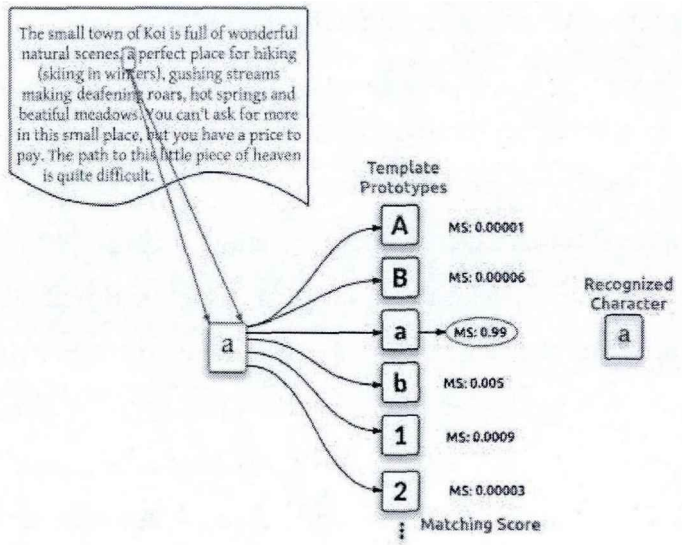


Fig 1.18 Explicit segmentation based methods. An individual language unit (a character) is matched against templates and recognized as a character where the matching score is the highest [6]

Initially, considerable efforts were devoted for explicit segmentation based approaches. Pal and Sarkar [21] tested their recognition system on 3050 characters by using features of contour and water reservoir. The system recognized characters and numbers with accuracy of 97.8%.

Megherbi et al. [22] used fuzzy logic for classification and defined seven classes of 36 Urdu characters. The authors relied on features that include the number of dots of a character, placement of dots, secondary components, aspect ratio and angle between initial and final points and achieved high accuracy. The implemented MLP Network [41] composed of input, hidden and output layer. The output layer contains 16 neurons for providing 16-bits of Unicode for each character. Individual character are recognized with an accuracy of 98.3%. A segmentation based approach [26] proposed for recognition of characters having 36 Noori Nastaleeq font size. Before applications of thinning, secondary and primary components are detached. After thinning, the ligatures are segmented at branching points.

Pathan et al. proposed system for recognition of 47 handwritten isolated character [45] on the basis of Invariant Moments. Urdu characters were grouped into single and multi-component characters. The single component characters are normalized and 28 Moment

Invariant features are calculated. For multi component character, primary and secondary component are detached. The primary component image is processed similar to single component characters. The secondary components are normalized and divided into 2 horizontal zones. Secondary strokes are counted with their types like bay (ﺑ)with one dot below, pey (ﭘ) with three dots below, sey (ﺛ) three dot above and te (ﺗ) one toe above. The reported accuracy is 93.59% isolated handwritten Urdu characters.

Tabassam Nawaz et al. [42] performed line and character segmentation after preprocessing and noise removal. The extracted characters stored in .xml file for training purposes. In recognition stage, characters are segmented and chain code is obtained. This code is then compared with the xml file already created. Authors claimed 89% recognition accuracy. Another segmentation based techniques is used by Ahmad et al. [40] for Nastaleeq font. A neural network is trained on an Urdu character data set which contains all forms of every character. The main steps include base line identification, words and character segmentation. These separated characters are then given to a trained Neural Network for identification and reported accuracy is 93.4%.

Tariq et al. [43] applied NN classifier for recognition for Urdu characters by using structural features and reported accuracy is 97.43%. Urdu Character Recognition using PCA is presented in [46] and the recognition accuracy obtained was 96.2 %.

A MLP networks is trained by back propagation algorithm [25]. The main advantage of Back Propagation Neural-network (BPN) method is that it can fairly approximate a large class of functions. Preprocessing step involves noise removal (by median filters) and binarization process. The system character recognition accuracy is 84%.

b) Implicit Segmentation based OCR

Implicit segmentation is also known as over segmentation approach [6]. This approach is applicable to languages and text where correct character segmentation is impossible. This may happen due to the overlapping, touching characters, non-availability of clear boundaries of characters in a document image.

Implicit segmentation methodologies necessitate labels for images of language units like characters, ligatures, words and text-lines [30]. By using imperfect basic language units/character are then recognized by standard machine learning approaches including KNN, neural networks, decision trees, Bayes classifiers etc. For approximating boundaries

between two neighboring characters, one can use [37] methods namely: (i) vertical projection-based methods, (ii) feature based methods, or (iii) thinning or skeleton analysis. A very well illustration of implicit (over) segmentation based methods is given in Fig 1.19. So far, all implicit segmentation based OCR are used for recognition of sentences with the help of LSTM architecture that can extract and learn contextual information along with the shape of language units.

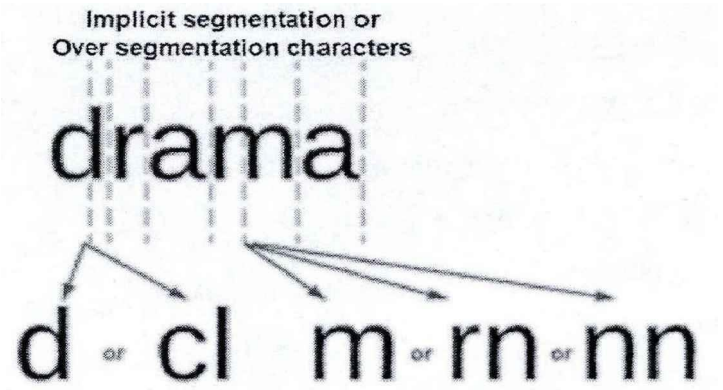


Fig 1.19 Example of Implicit segmentation based methods. The segmentation points are approximated and each segment is recognized as a character with a confidence score. The most probable character after the other is estimated by a language model. [6].

Recurrent neural network (RNN) is successfully used to model context in sequential type of data like text and speech. So, this characteristic of RNN makes it appropriate for the development of context sensitive OCR systems for printed Nastaleeq scripts [48]. The authors have applied Bidirectional Long Short Term Memory (BLSTM) architecture with Connectionist Temporal Classification (CTC) layer at output to recognize printed Urdu text. They evaluated BLSTM networks for two cases: one without considering character's shape variations and second by considering them. The recognition error rate for first and second case are 5.15% and 13.6%, respectively. These results were obtained on synthetically generated UPTI dataset containing artificially degraded images to reflect some real-world scanning artefacts along with clean images.

Naz et al. [29, 49, 50, 52, 53] used implicit character segmentation with sliding window for feeding Multi-Dimensional Long Short Term Memory (MDLSTM) with CTC at output layer for recognizing 43 classes of characters, achieving accuracies in the range of 93.38% to 98%. In all of these valuable contributions, authors have employed same split

of UPTI dataset while with different features at the input like statistical, zoning, raw pixel values, and convolutional.

Ahmed et al. [51] used implicit character based segmentation using sliding window method for feeding characters as input to BLSTM network with raw pixel values of super set sentences of UPTI dataset . This system also recognizes 43 classes of Urdu characters with an accuracy of 88.94%.

1.5.2 Segmentation Free OCR

Segmentation-free or holistic OCR approaches are different from the segmentation-based on the basis of segmentation level required. Conventionally, such approaches try to recognize words or sub-words (ligature) for a given text. Such approaches extract features for a word, sub-word, or ligature. These features are then used to train a classifier to recognize the whole word or complete ligature.

A general OCR system [59] named Nabocr, is proposed for Arabic script languages. Nabocr defined OCR approaches related to Arabic script recognition. Nabocr recognizes both Urdu and Arabic text in Nastaleeq and Naskh fonts, respectively. It can be used for other Arabic scripts. In order to evaluate Urdu recognition, UPTI (Urdu Printed Text Image) database is used. The performance of the system for Arabic and Urdu text are 91% and 86%, respectively. Moreover, the performance of Nabocr is compared to Tesseract's Arabic recognition module and found almost same recognition accuracy.

The framework of Urdu Nastaleeq OCR [60], which recognizes Urdu Nastaleeq document images having specific font size (from 14 to 44). In preprocessing document image is subdivided into text and non-text areas. Latin and Nastaleeq texts are separated in this step. In classification and recognition phase, Latin text is recognized by Tesseract open source system. Ligatures of Nastaleeq text are recognized by two ways. First the tailored Tesseract engine is used for recognition of Nastaleeq ligatures' main bodies. Second method uses HMM with DCT features for segmented parts of main ligature body. Both classifiers recognize the main body of ligature with rank. After recognition text is converted into best words and best sentences. The claimed ligature recognition accuracy is 86.15% when tested on 244 document images.

HMM based font independent Urdu OCR is devised by [24]. In which the Urdu text images are normalization and segmented up to ligature level. DCT for parts of ligatures are calculated and recognition is done using HMM with an accuracy of 97.8%.

Hidden Markov Model based OCR model [47] recognizes the corresponding ligature from taking the main body (without diacritics) as input and reported accuracy is 92.73%.

Font size independence for Noori Nastaleeq OCR is presented in [61] which is very much similar to the work of [47]. The system extracts outlines of input ligatures and subsequently normalized to acceptable font size for feeding to system. The recognition rate is 98% for manually generated data and 96% for scanned books and magazines.

Segmentation free technique is used in [15]. The authors used HMM model with DCT feature vector for the recognition of main body of ligature. Also, a feature vector was created for identification of diacritics. The claimed accuracy is 92%. The recognition process of [36] recognizes ligature with 94% accuracy for Urdu Nastaleeq font size of 36.

1.6 Motivation

The goal of this study is to develop techniques that can recognize Urdu text on the basis of ligatures. During the last decade, Urdu Nastaleeq OCRs have been successfully implemented based upon characters [21, 22, 25, 40-43, 45, 46], ligatures [15, 24, 36, 47, 59-61], and sentences (context based) [30, 48-53] as recognition unit. Recently, systems based upon ligatures and sentences (context based) proven more accurate as compared to character based OCR systems. However, there are three main concerns that are required to be addressed in order Urdu OCRs to be more accurate. First, the accuracy of Urdu OCR systems is directly linked with accurate segmentation of page into text lines and ligatures. Secondly, Urdu OCRs are trained on the basis of characters, ligatures, and sentences. Specially, ligature and context based OCR systems are proven more successful. However, the ligature based systems are trained on a very small data sets. The largest dataset used so far consists of approximately 83000 ligatures [60], which is not enough for evaluating commercial Urdu OCR. Therefore, it will be more appropriate to use a large amount of training data for the purpose of training OCR systems for practical use. Furthermore, many practical OCR applications necessitate automatic feature extraction process instead of human crafted features. Human crafted features are proven to be more error prone and don't represent every aspect of underlying image or data. Here for Urdu OCR, it is also

required to automate the process of feature extraction without human involvement. This study also aims to put forward a ligature based RNN instead of character based methods that are very difficult to label.

1.7 Contribution of Thesis

The major contributions of this thesis are in the field of Urdu Nastaleeq OCR and are detailed as below:

1. The recognition accuracy of ligature based Urdu language Optical Character Recognition systems highly depends on the accuracy of segmentation that converts Urdu text into lines and ligatures. Segmentation of Urdu text pages into lines and ligatures has been improved. Classical horizontal projection based segmentation method is augmented with a propose Curved-Line-Split algorithm for successfully overcoming the problems such as text line split position, overlapping, merged ligatures and ligatures crossing line split positions. Ligature segmentation algorithm extracts connected components from text lines, categorizes them into primary and secondary classes, and allocates secondary components to primary by examining width, height, coordinates, overlapping, centroids and baseline information.

2. UPTI dataset segmented for using it for experimental purposes to overcome the problem small datasets used for training in prevalent work. UPTI dataset is segmented into 189000 ligatures from 10063 text lines having 332000 connected components with accuracy rate of 99.80%.

3. Prevalent Urdu OCRs are mostly relying on human crafted features that are proven more error prone and don't represent every aspect of underlying image or data. Here, autoencoder and stacked autoencoders are used to extract features directly from raw pixel values and resulted in superior results than the prevailing contributions.

4. A novel LSTM framework is introduced for context based Urdu sentence recognition. This frame work is named as Gated BLSTM (GBLSTM). It combines the output of both BLSTM layers and select only the common information for output. Furthermore, GBLSTM is ligature based as opposed to the earlier contributions where character based LSTM were used. As, it is easy to label sentences with the help of ligatures instead of characters. As, correct characters labeling needs considerable human labor.

1.8 Thesis Structure

The focus of the thesis is the applicability of deep learning approaches for Urdu CR. It mainly deals with segmentation, feature extraction and recognition of Urdu Nastaleeq text. The rest of the thesis is divided into six chapters. The organization of the thesis is as follow:

This chapter 2 summarizes the preliminaries of Urdu OCR techniques. Moreover, the underlying chapter briefly presents basic text segmentation techniques with their limitations. Further, Artificial neural network is discussed in detail. The employed ANN methods are described with their mathematical notations. The evaluation deep learning methods and the latest research scenarios are discussed in details. Also, datasets for Urdu text are tabulated for better understanding.

The chapter 3 puts forward two effective line and ligature segmentation algorithms for Nastaleeq Urdu document images. For this purpose, the projection profile method with a novel Curved Split Line algorithm is explained for segmentation of text lines of Urdu document images in detail with algorithms. Further, a ligature segmentation algorithm is explained that successfully segmented text lines of UPTI dataset with an accuracy of 99.08%. Overall, both proposed segmentation algorithms performed comparable with the state of art segmentation algorithms for Urdu document images.

Chapter 4 proposes a novel mechanism for ligature recognition while extracting features by stacked denoising autoencoder with softmax classifier. For this purpose, features are extracted using different SDA networks and best network is chosen for testing purpose. Same data is also tested on SVM. Overall, the proposed URL-SDA learning models outperformed the SVM and prevailing models for Urdu OCRs.

Chapter 5 proposes a ligature based gated BLSTM approach for sentence recognition. The prevalent contribution for Urdu sentence recognition have used only character based methods. Such systems contain very difficult process of labeling of text lines with appropriate shaped characters. On the other hand, labeling text lines with ligatures is comparatively easier. Further, GBLSTM methodology provided improved result because of combining information of forward and backward layers appropriately with the help of multiplicative gate layer. The performance of GBLSTM is comparable with the earlier contributions using multidimensional LSTM architectures.

Eventually, Chapter 6 concludes the dissertation with some possible extensions, suggests further investigation and recommendations, and highlights future directions of the Urdu OCR domain.、

CHAPTER 2

Preliminaries

2.1. Projection Profile Method

This method is a histogram based that accumulates number of black pixel values parallel to pixel rows of an image [64, 66]. One can construct such projection profile at any angle. Generally, it is represented horizontally or vertically along pixel rows or pixel columns respectively. Such horizontal and vertical representation are called Horizontal and vertical projection for a document image, as shown in Fig 2.1.

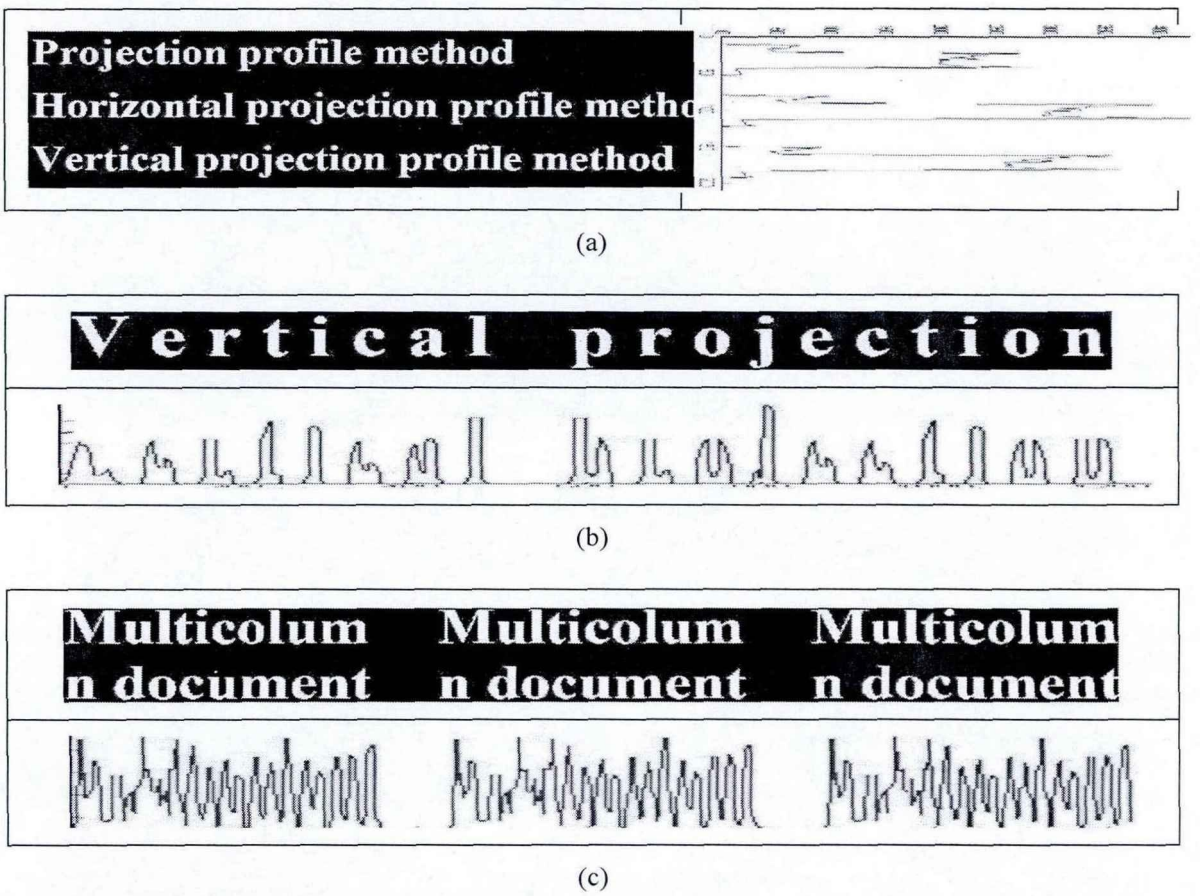


Fig 2.1 Projection profile method, (a) Horizontal projection profile representing text lines, (b) Vertical projection profile method representing characters, (c) Vertical projection profile method representing multiple columns

In a document image with horizontal text lines, the width of horizontal projection profile peaks are equal to the character height and valleys represents interline spacing as depicted in Fig 2.1(a). The characters may be spotted in a sentence image by vertical projection, where each peak represents a character as shown in Fig 2.1 (b). For a document, each column has a peak in vertical projection profile, troughs represent inter-column spacing. Projection profile methods are used for text line segmentation [65], skew estimation [67], layout segmentation [68], and in many different ways. For a document image containing 'm' rows and 'n' columns [69], the horizontal and vertical projection are given below.

$$\text{Horizonal Projection(x)} = \sum_{y=1}^{n} f(x, y) \tag{2.1}$$

$$\text{Vertical Projection(y)} = \sum_{x=1}^{m} f(x, y) \tag{2.2}$$

2.2 Salt-and-Pepper Noise (SP)

Image capturing process may add unwanted signals to resultant image that adds false and unnecessary information. One of such noises is salt-and-pepper noise that is caused by sharp and unexpected changes in the image signal and is also called as impulse noise. It is like white and black pixels scattered sparsely on the image. It can be formed by dust contaminated on camera lens, sensors and malfunctioning circuitry of a scanner or digital camera. For reducing either salt noise (unwanted white pixels) or pepper (unwanted black pixels) noise median filter [78], a morphological filter [79] and contra harmonic mean filter [80] are used in literature.

2.3 X-Y Cut

A projection profile based top down page segmentation method known as X–Y cut [81]. This approach segments a document image in to a number of rectangular segments. This method finds the widest horizontal and vertical empty rectangles/valleys. The underlying page is then segmented based upon the horizontal/vertical empty rectangles. The same segmentation strategy applies recursively to each segment/block and stops when no empty block can be found greater than a set threshold size. Instead of using image pixels, black pixels of bounding boxes for connected components [82] can be used. This

way, the recursive XY-cut become much faster. Figure 2.2 demonstrates the XY-cut method for the segmentation of a page image.

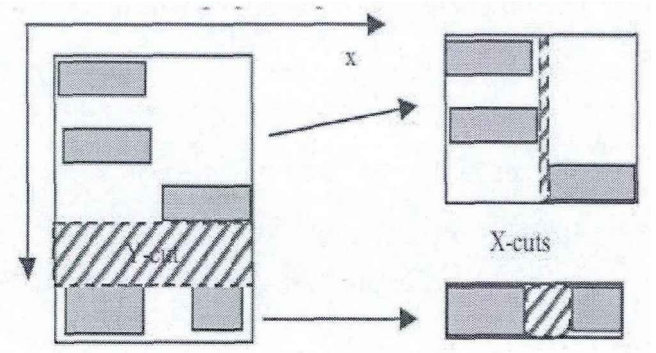


Fig. 2.2 Gray segments represent some bounding boxes. The lined segments indicate the widest empty valleys where to segment. The left and right pictures show the horizontal and vertical segmentation respectively [81].

Segmentation and reading order will become a problem if there are L-shaped segments or block. This problem is dealt by Jean-Luc Meunier [83] as shown in Fig. 2.3.

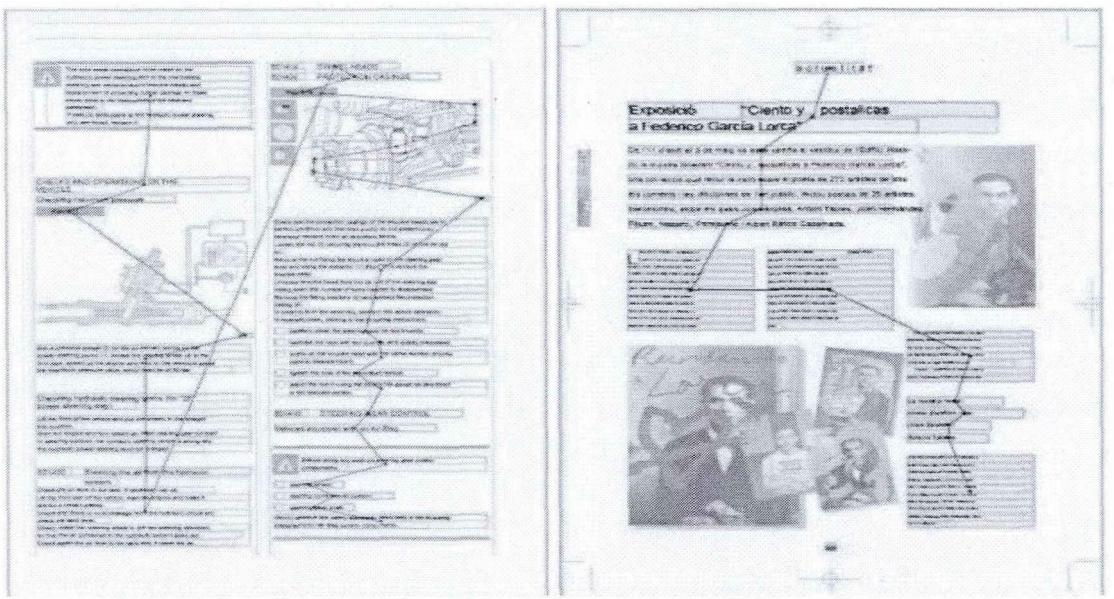


Fig. 2.3 Segmentation reading order by Optimized XY-Cut [83]

2.4 Smearing Algorithm

The smearing algorithm [84] deals with binary images where 0 and 1 represent white and black pixels, respectively. If the number/width of adjacent 0's is less than a threshold T+1 then these algorithms converts sequences of 0's in x to sequences of 1's in y.

These method merges the adjacent black segments where less than T pixels separates them. These algorithms are applied row and column wise to the document using thresholds t_{sh} and t_{sv}, respectively, resulting in two unique bitmaps. The resulting bitmaps are merged using logical AND operation. By using a smaller threshold, tsm another horizontal smearing is applied for more smoothness of resultant bitmap. Subsequently, connected component method is applied to get the document segments from resulting bitmaps. Next step is to calculate mean horizontal run-lengths of the information\black pixels (Rmean) and the mean block height (Hmean) from the original image. A block is classified into a text block [85] if:

$$HRun < ftr.Rmean \quad and \quad BH < fth.Hmean \tag{2.3}$$

Where, ftr and fth are two thresholds, HRun is the horizontal run-length of the black pixels in the current block, and BH is the block height.

2.5 Rectified Linear Unit (ReLU)

The rectified linear (ReL) is a nonlinearity function that may be used as a replacement of sigmoidal function. ReLU overcomes the limitations of sigmoidal function [86]. Mathematically, ReL function can be written as follows:

$$\mathrm{hAct}^{(i)} = \max\left(\mathrm{Weight}^{(i)T}\mathrm{x}, 0\right) = \begin{cases} weight^{(i)T}x & if\ weight^{(i)T}x > 0 \\ 0 & otherwise \end{cases} \tag{2.4}$$

The partial derivative of ReL unit is 1 when it is activated above 0. Thus, in an arbitrarily deep network, vanishing gradients problem can not affect the active hidden units. Furthermore, the saturation point for ReL units is exactly 0 and this is particularly helpful when using the respective hidden units for feeding input to classifiers.

2.6 Softmax Function

Softmax function is a generalization of logistic function. It applies to the problems where class for a given instance may be from a sequence 1 to some value y.

2.7 Artificial Neural Network (ANN)

An artificial neural network comprises two sets Nodes(N), Edges(E) and a weight function EW, where N is a set of neurons and E is a set {(i,j) such that i,j $\in$ N} represents edges\connections between neurons i and j

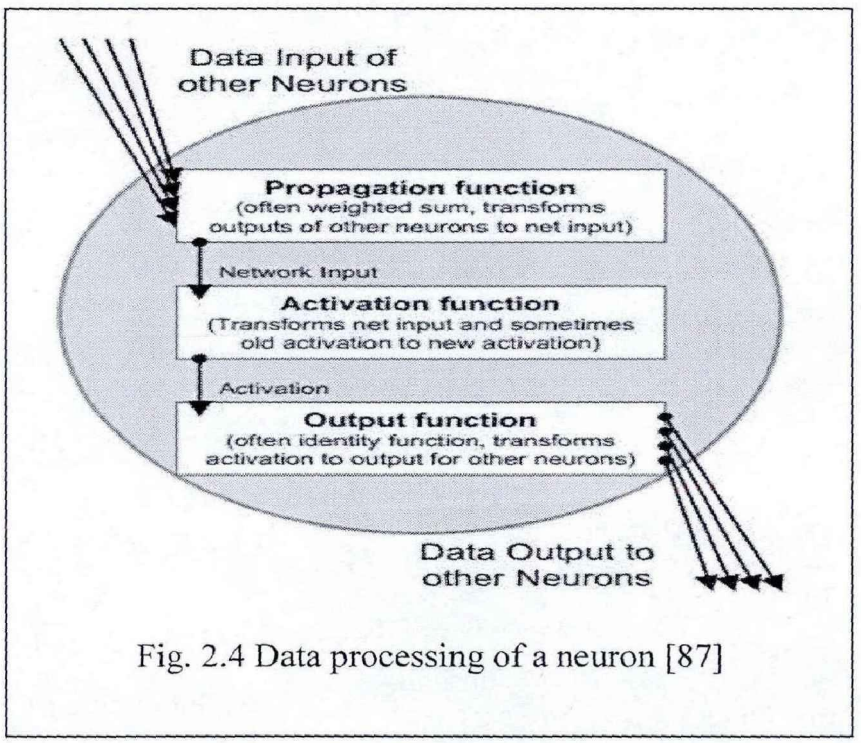


Fig. 2.4 Data processing of a neuron [87]

.The weight function EW: E $\rightarrow$ R, where $W_{i,j}$ denotes the weighted link of neuron i and j. When there is no connection between i and j neuron then its value either be 0 or undefined.

The propagation function for a neuron j

The outputs $O_{i1}, \ldots, O_{in}$ of all connected neurons to jth neuron are given as a input to the propagation function by the edges containing weights $W_{i1,j}, \ldots, W_{in,j}$, and transform into a form that can be easily handled by an activation function. Therefore, a neural network input is there because of the propagation function. The weighted sum is a very popular propagation function given as follows:

$$\text{net}_j = \sum_{i \in I}(o_i \cdot w_{i,j}) \quad (2.5)$$

A neuron may be activated or excited based upon the neuron j's previous activation state and current external input. A neuron gets activated if its input exceeds certain threshold. The input is checked against the threshold value by a function known as activation function, to make the neuron activated. It can be represented as follows:

$$a_j(t) = f_{act}(\text{net}_j(t), a_j(t-1), \theta_j) \quad (2.6)$$

Where, θ is a threshold value.

2.7.1 Common Activation Functions

The binary\heaviside threshold function is the simplest activation function. The function changes its value if the input exceeds a certain threshold value otherwise remains constant. Also another very useful activations function is the logistic\fermi function and it maps the input values to (0, 1). It can be represented as follows:

$$\frac{1}{1+e^{-x}} \quad (2.7)$$

The hyperbolic tangent is another activation function that maps input values to (−1, 1). The logistic function can be expanded by a parameter 'T' known as temperature parameter into the form given below:

$$\frac{1}{1+e^{\frac{-x}{T}}} \quad (2.8)$$

The smaller value of this parameter will compress the function more on the x axis and implicitly it approximate the Heaviside function. All three functions are shown in Fig.

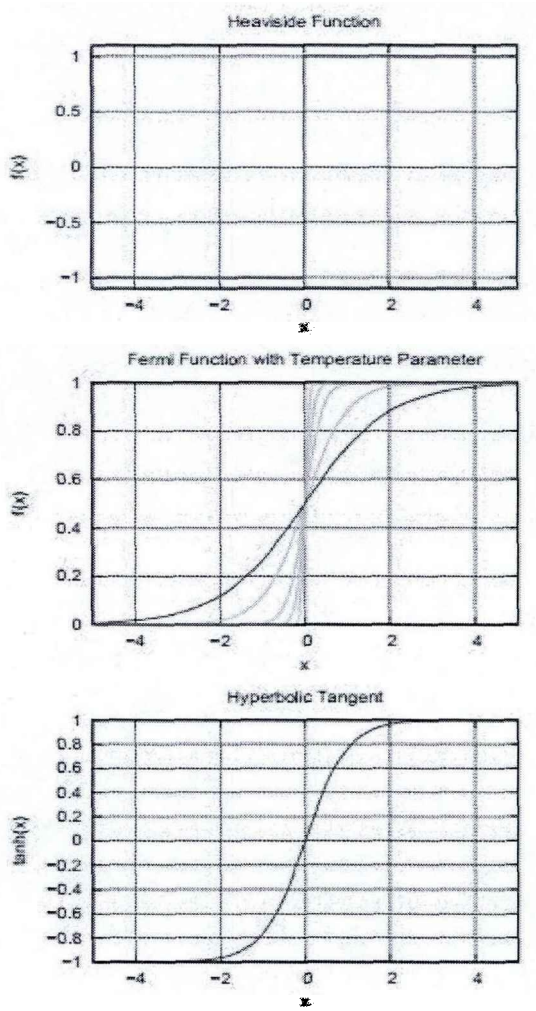


Fig. 2.5 Activation functions [87]

A neuron j will need to prepare some new value to be forwarded to its connected neurons after activation. This is calculated by output function and is represented as follows:

$$f_{out}(a_j) = o_j \tag{2.9}$$

2.7.2 Network Topologies

Physical layout of neurons in to a network is known as network topology. Generally topologies contains one input and one output layer with n hidden layers. Some of the topologies are as under:

a) Feedforward: Each neuron of previous layer will only the possibility of directed connections to the neurons of the next layer (towards the output layer).

b) Feedforward network with shortcut connections: A special case of feedforward network in which the connections may be directed towards any next subsequent layer.

c) Recurrence: The output of a neuron influence itself or previous layers' neurons, usually with the help of a connection.

d) Direct recurrences: It is a type of recurrence in which a connection starts and ends at the same neuron. Here, a neuron j connects to itself, with the weight $w_{j,j}$ and such connections are represented by diagonal of weight matrix.

e) Indirect recurrence: A network of neurons where connections between neurons and their preceding layers being allowed.

f) Lateral recurrence: Such recurrent network permits connections to any neuron within the same layer.

g) Complete interconnection: Every neuron is connected to every other neuron in such topology.

2.7.3 Learning Paradigms

Learning paradigms refers to the algorithms and techniques with the help of which a neural network learn by modifying the weights of connections without changing the topology of network.

a) Unsupervised Learning: In unsupervised learning the training set only contains input patterns without output. Therefore, the neural network finds out the class of the input data based upon the similarity measures. A popular example of such algorithm is Kohonen's self-organising maps.

b) Reinforcement Learning: In this type of learning in which a training set contains of only input examples\patterns. The network executes and outputs a value about the validity of the result. So, the network receives reward or punishment based upon with output.

c) Supervised Learning Methods: Such methods provide input training patterns with expected outputs so that the network can calculate error and update weights accordingly. Generally, such methods consist of following 5 steps.

i) Entering: Input is given to the network.

ii) Forward propagation: Output is generated.

iii) Comparing: The output and desired value are compared and an error vector is prepared.

iv) Corrections is calculated based on the error vector.

v) Corrections are applied.

2.7.4 Perceptron

A perceptron is a processing unit that takes input through weighted links and based upon a threshold produces a binary output [88]. Generally, one can think of a perceptron as a feedforward network containing the first weighted layer for input. It is followed by at least one trainable weight layer. The previous layer is completely linked with the following layer. The perceptron[1] was the precursor to the backpropagation based artificial neural network model.

A Single Layer Perceptron (SLP) [88] is a feed forward network containing only one layer of trainable weights with an output layer.

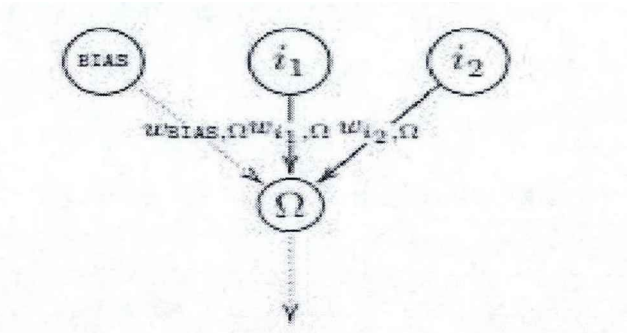


Fig. 2.6 A single layer perceptron with two input nodes and one output neuron [87].

[1] https://en.wikipedia.org/wiki/Perceptron

A single layer perceptron is capable of performing boolean AND, OR and NOT operations. A single perceptron can classify the data into classes when it is linearly separable as shown in the following Fig. 2.7.

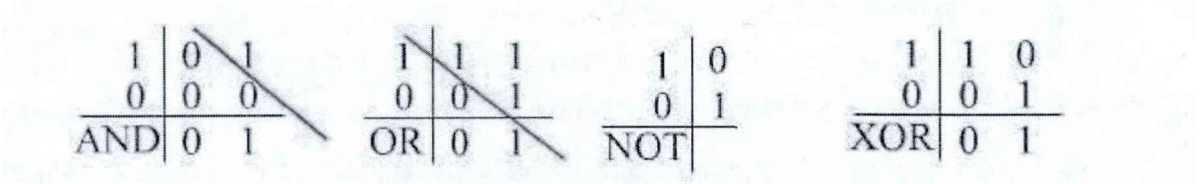


Fig. 2.7 AND, OR and NOT logic functions are linearly separable while XOR logic function is not a linearly separable function.

One perceptron layer can draw a line for demarcating two classes. In case of XOR operator that is not linearly separable and so single perceptron can't separate two classes. This problem could be solved by more than one perceptron employed in a feed-forward network.

2.7.5 Multilayer Perceptron

A feed forward network containing with more than one layers of perceptrons with trainable weighted connections are known as Multi-Layer Perceptron (MLP) [89]. A MLP is an n-layer feed forward perceptron neural network having exactly n variable weight layers and $n+1^{th}$ layer is the input layer. A MLP contains more than one trainable weight layers. An MLP with a single hidden (trainable weighted) layer is represented in Fig. 2.8 as follows:

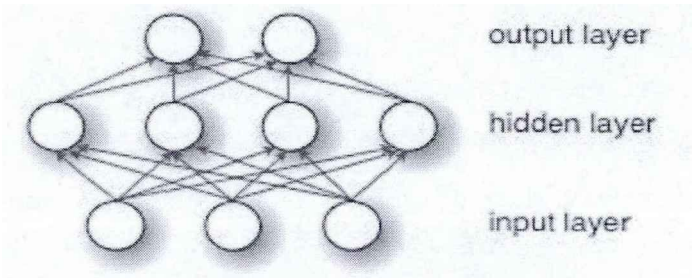


Fig. 2.8 A simple Multilayer perception[2]

[2] http://deeplearning.net/tutorial/mlp.html

2.8 Autoencoder

Feature learning is the most well-known use of deep neural networks [71-73, 74]. It can learn useful feature for a given data [75]. It may learn a compact representation with reduced dimensions of the given data [72].

Autoencoder is basically a feed-forward and non-recurrent neural network [70]. There may be one or more hidden layers in between input and output layer as shown in Fig 2.9. An autoencoder is a MLP except that it has equal number of inputs and output nodes. Moreover, an autoencoder predicts the input value x at the output. It does a feed-forward pass and predicts the value of$\bar{x}$, measures the difference between x and $\bar{x}$ and back propagates by performing weight updates.

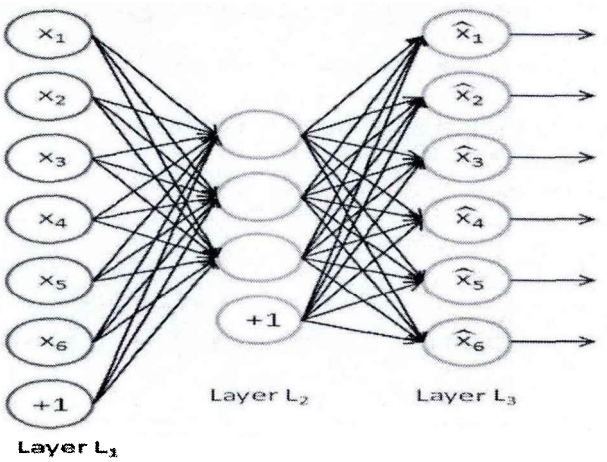


Fig 2.9 Basic structure of autoencoder, number of inputs and outputs are equal, and out $\hat{x}_i$ is approximation of x_i, layer 2 is the code layer representing the compact representation of input[3]

If the hidden layers are less than the input/output layers, then the final hidden layer represents the compressed representation of the input [76]. As mentioned earlier, autoencoder is a subtype of MLP, therefore it can use all the activation functions used in MLP.

3

An autoencoder takes the input $\mathbf{x} \in [0, 1]^d$ and encode it to a middle layer representation $\mathbf{y} \in [0, 1]^{d'}$ e.g.:

$$y = act(Wx + b) \tag{2.10}$$

Where act may be any nonlinear activation function e.g., sigmoid. The encoded y is used to reconstruct the approximation z of the original x as follows.

$$z = act(W'y + b') \tag{2.11}$$

Where, $W'$is W^T.

The parameters' optimization is incorporated by minimizing mean error during reconstruction.

2.8.1 Denoising Autoencoder

Denoising Autoencoder is a type of autoencoder that tries to learn the input x at the output as $\hat{x}$ by working on partially corrupted version of input. In this way, the learnt $\hat{x}$ may be more stable, robust and gives better higher level representation [70]. Different methods are available for corruption of input data, like MN, a suitable proportion of image pixels are masked to 0; Salt-and-Pepper Noise (SP), where randomly selected pixels of an image are set to maximum and minimum values of the image uniformly [18]. We have used MN corruption method in our experiments.

2.8.2 Stacked Denoising Autoencoder

A deep network can be formed be stacking denoising autoencoders and by passing the intermediate code representation of previous layer to next layer [70]. Every layer is considered as a separate denoising autoencoder and trained by minimizing error incurred during approximation of this layer's input. The output of every preceding layer will be considered as the input for next coming layer, while first layer gets input from input vector. In this way, we can train k+1 layer only when we have already trained the all first k layers. After pre-training, fine tuning is applied in a supervised way to minimize prediction error

by adding logistic layer at the top of network. A network is formed by all encoding parts of each auto-encoder and subsequently trained as a multilayer perceptron.

2.9 Hidden Morkov Model

Hidden Markov model (HMM) is a probabilistic model for linear sequential learning problems. Such models are known as sequence labeler or sequence classifier. Every unit of sequence is allotted a label or class by such models. Markov Chain is a weighted automaton that determines what states will the used based upon the input sequence. The problems with ambiguities can't be represented by this model. It can be considered as a conceptual toolkit for modeling complex problems just by sketching their instinctive picture. HMM has become the heart for many application including multiple sequence alignment, profile searches, gene finding and monitoring site identification. The detailed tutorial can be read from [77].

2.10 Support Vector Machines

Among successful supervised machine learning based binary classifier, Support Vector Machines (SVM) have got more popularity because of its efficient learning of data and patterns [90]. More precisely, SVM model builds an optimal hyper plane with largest functional margin for an effective separation of examples of two dissimilar classes. Moreover, it can handle linear and non-linear data. Given a set of tuples with n training examples ((X1, yi), (X2, yi), ... , (Xn , yi)), where yi $\in$ {−1 , +1}, each yi has a corresponding class depending upon feature vector Xi. The idea here is to build a hyperplane with maximum functional margin that can bisect the vectors Xi on the basis of output values +1 and -1, while reducing the classification error. The resulting hyperplane must satisfy the input X according to the following equation:

$$W^T.X_k + b = 0 \qquad (2.12)$$

Where, w is the weight vector and b is the bias. The values of w and b are updated during training of SVM. Furthermore, the offset of origin to the hyperplane is governed by rule b/(||w||). Thus, it is essential to solve the following primal problem to get the solution of binary classifier.

$$\min(||w||) \text{ , subject to } yi(W^T.X_k + b) \geq 1 \text{ , for } k = 1, \cdots, n \qquad (2.13)$$

2.11 Recurrent Neural Network

Human beings have the capability to use the prior experience for performing different tasks. Conventionally, neural networks don't have the property of using prior learning and this is the major limitation. This problem is addressed by recurrent neural networks (RNN). These networks have the connections from succeeding layers to preceding layers or having loops that allows the networks to input the previously given information to it as input.

A RNN can be considered as multiple replicas of the same neural network, each handing over a message to a successor. A RNN with its unrolled version is shown in Fig. 2.10 given below.

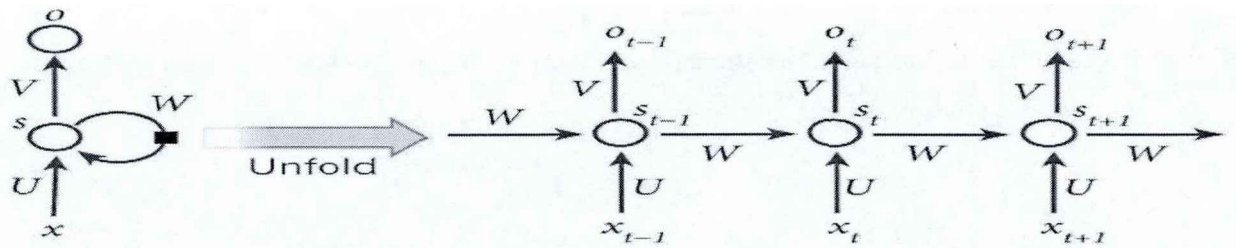


Fig. 2.10 A RNN on the left and its unfolded form on the right side[4].

At time stamp t, x_t *and* s_t are the input and hidden state of the above shown RNN. The output of the previous state and current input are used to calculate the state s_t.

$$s_t = f(Ux_t + Ws_{t-1}) \qquad (2.14)$$

The activation function f usually be tanh or ReLU. s_{t-1} for first state will be zero. The output o_t at time t is calculated as:

$$o_t = f(V.s_t) \qquad (2.15)$$

From last decade, RNNs have been applied for solving many problems: speech recognition [91], language modeling [92, 93], translation [94], and image captioning [95] etc. The RNN has achieved much popularity because of its special form known as Long

[4] http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/ April 15,2017

Short Term Memory architecture. The RNNs are incapable to learn "long-term dependencies" [96] because of the vanishing gradient problem.

Bidirectional RNNs are is a kind of RNN where the output at time stamp t depends on the previous elements and also the future elements in the sequence. Bidirectional RNNs are two stacked RNNs. On the basis of both RNNs, the output is constructed. For example, to insert a proper word in a text for a missing one, one should look for both context (before and after that word) before prediction.

2.12 Long Short Term Memory

Long Short Term Memory network (LSTM) is a special type of RNN, having the capability to remember long-term dependencies. LSTM was first presented by Hochreiter & Schmidhuber [97] and nowadays it is extensively used model in Deep Learning for NLP. They are being used for performing many learning task with outstanding performance. LSTMs are specifically designed learn long-term dependency problem that was the limitation of simple RNN.

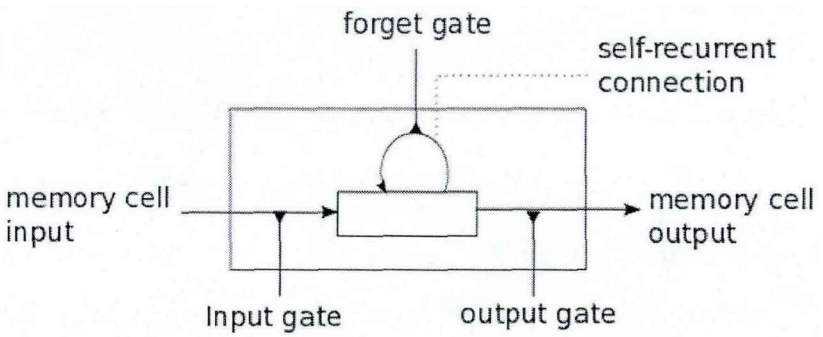


Fig. 2.11 An illustration of LSTM cell[5], f=forget gate, memory cell output=c_t, memory cell input=c_{t-1}, input gate=i, output gate=o

The multiplicative gate units allow LSTM to remember information for a long period of time, therefore overcome the problem of vanishing gradient problem. The LSTM architecture consists of a set of recurrently linked memory blocks, named memory cells. These cells can store information like memory chips in a computer. Each cell contains at least one connected memory cell with input, forget and output gates. These gates are used

[5] http://deeplearning.net/tutorial/lstm.html

for maintaining the cell state. LSTMs solves vanishing gradients problem through this gating apparatus. A hidden state s_t of LSTM can be calculated as follows:

$$i = \sigma(x_t U^i + s_{t-1} W^i) \tag{2.16}$$

$$f = \sigma(x_t U^f + s_{t-1} W^f) \tag{2.17}$$

$$o = \sigma(x_t U^o + s_{t-1} W^o) \tag{2.18}$$

$$g = tanh(x_t U^g + s_{t-1} W^g) \tag{2.19}$$

$$c_t = c_{t-1} \otimes f + g \otimes i \tag{2.20}$$

$$s_t = \tanh(c_t) \otimes o \tag{2.21}$$

$Where, W^i, W^g, W^f, W^o, U^i, U^g, U^f, U^o$ are weight matrices.

2.12.1 Bidirectional Long Short Term Memory

Bidirectional LSTM (BLSTM) is used to draw the context or sequence information from both the past and the future as shown in Fig. 2.12. For this purpose, two LSTM layered network known as bidirectional LSTM model, is used to remember the preceding and following context information inside sequential data. One LSTM layer encodes the text\sentence from start to end, and the other in opposite direction. Then the output of both layers are at specific time stamp t is concatenated to get the intermediate representation.

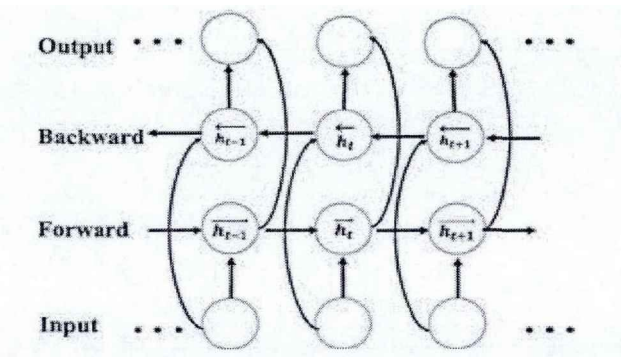


Fig. 2.12 An illustration of bidirectional LSTM [98]

2.13 Connectionist Temporal Classification

Connectionist Temporal Classification (CTC) is specifically designed for sequence data labelling when there is no alignment between the inputs and corresponding labels. It does not require pre-segmented input and its corresponding labels. CTC does this by label predictions for the input sequence. This overcomes the problem of segmentation of input data. Further, CTC itself finds probabilities of all labels in a sequence.

A CTC layer is used to label the data by using a softmax output layer (Bridle, 1990) for L+1 probabilities for L labels. The |L | units are taken as the probabilities of occurrence of the labels at specific times and extra one unit is used to represent the probability of occurrence of a 'blank' or no label. CTC is very useful for languages with cursive scripts, speech data, and handwritten data where segmentation is very difficult task.

2.14 Datasets

The detail of the available Urdu text image data set is given in the following Table 2.1.

Table 2.1 Available data sets for Urdu Text Images

S. No.	Name of Urdu Text Image Data Sets	Designers
1	Urdu Printed Text Images (UPTI)	Sabbour et al [59]
2	Center for Pattern Recognition and Machine Intelligence	Sagheer et al. [99]
3	Center for Image Processing-Urdu Corpus Construction Project (CENIP-UCCP)	Raza et al [100].
4	CLE Urdu HFL with point sizes 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, Document Images containing 5586 classes.	cle.org.pk
5	CLE Urdu HFL 14, 16, 18, 28 Point Size Instance Images with 5586 classes.	cle.org.pk
6	CLE Urdu Image Corpus 14,16 Point Size	cle.org.pk

7	CLE Urdu 14, 16 Point Size Instance Images	cle.org.pk
8	CLE Urdu HFD 14, 16 Point Size Instance Images	cle.org.pk
9	CLE Urdu 14,16 Point Size Diacritic Instance Images	cle.org.pk
10	CLE Urdu 14,16,18 Point Size Distorted Instance Images	cle.org.pk
11	Urdu Nastaliq Handwritten Dataset (UNHD)	https://www.researchgate.net/project/Urdu-Handwritten-Research-UCOM-UNHD

2.15 Project

Center for Language Engineering (CLE)

(http://www.cle.org.pk/).

Pakistan is a multilingual and multi-cultural country where about 70 languages are spoken. More than half of these languages are written in Perso-Arabic Nastalique and Arabic Naskh styles. Gujarati, Gurmuki and Tibetan scripts are also used by some communities. Many communities are in search of suitable script to preserve their culture. All these languages have diversity in sounds and linguistic structures. Most of these languages still waiting for researchers for the preservation of disappearing languages and their cultural values.

One the most important work is started by Center for Language Engineering (CLE) for uplifting the research and development in linguistic and computational aspects of Asian and specifically Pakistani languages. The center is very active for conducting seminars, workshops and conferences for promotion of language processing nationally and internationally.

A special aim of CLE members is the development of Urdu language. The members of this center have published many research articles and theses. They have also created many different datasets for helping Urdu researchers with their needs of data.

The foundation of commercial Urdu OCR system was laid down by Center for Language Engineering. Uploaded Urdu OCR is working with the claimed accuracy of 86.15%. The authors are using two recognition methods at the same time for Urdu ligatures. It works for scanned pages with DPI300.

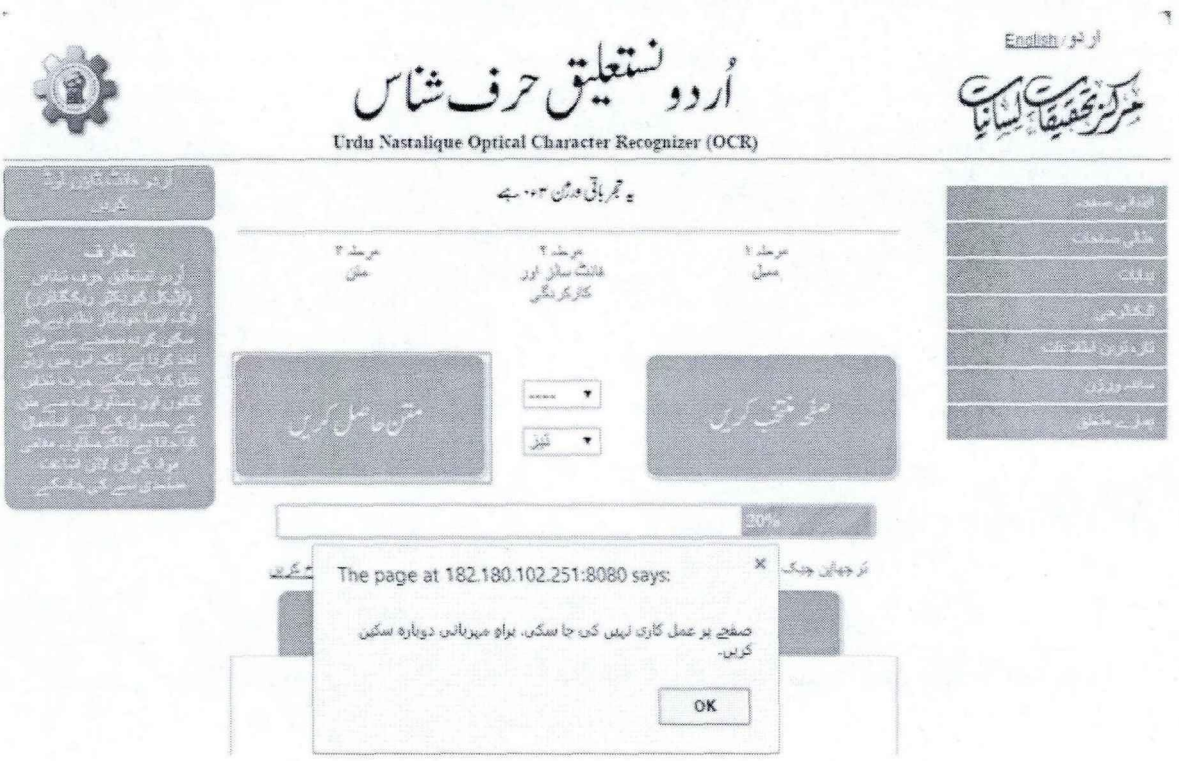


Fig. 2.13 Urdu Nastaleeq OCR from CLE [98]

2.16 Accuracy Measurement

Performance of the OCR systems are measured by recognition rate or accuracy that is calculated on the basis of number of correctly predicted ligatures as follows:

$$Accuracy = 100 * (1 - (W_{PL}/T_{SL})) , \quad (2.22)$$

Where,

W_{PL} = Wrongly predicted ligatures , T_{SL} = Total ligatures in test set

2.17 Summary

This chapter summarizes the preliminaries of Urdu OCR techniques. Moreover, the underlying chapter briefly presents basic text segmentation techniques with their limitations. Further, Artificial neural network is discussed in detail. The employed ANN

methods are described with their mathematical notations. The evaluation deep learning methods and the latest research scenarios are discussed in details. Also, datasets for Urdu text are tabulated for better understanding.

CHAPTER 3

Segmentation for Printed Urdu Text

The recognition accuracy of ligature based Urdu language Optical Character Recognition systems highly depends on the accuracy of segmentation that converts Urdu text into lines and ligatures. The ongoing research work on line and ligature based Urdu Nastaleeq recognition systems motivated us to put forward more accurate segmentation algorithms for Urdu Nastaleeq document images. Generally, lines and ligatures based Urdu language OCRs are more successful as compared to characters based. This chapter presents the techniques for segmenting Urdu Nastaleeq text images into lines and subsequently to ligatures.

This chapter details about the line and ligature segmentation algorithms. The line segmentation algorithm used classical horizontal projection based segmentation method is augmented with a Curved-Line-Split algorithm for successfully overcoming the problems such as text line split position, overlapping, merged ligatures and ligatures crossing line split positions. The proposed ligature segmentation algorithm is also discussed in detail that extracts connected components from text lines, categorizes them into primary and secondary classes, and allocates secondary components to primary by examining width, height, coordinates, overlapping, centroids and baseline information.

3.1 Introduction to Segmentation and Challenges for Robust Urdu OCRs

Urdu is the national language of Pakistan, and is widely spoken and understood in India and other Southern Asian countries as well. It is one of the top ten influential

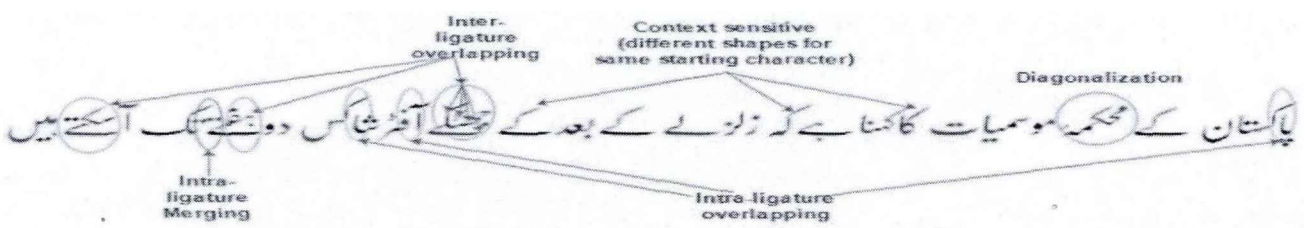
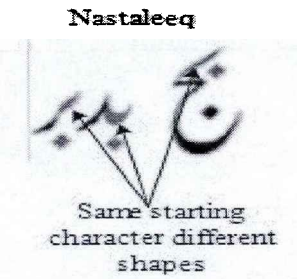


Fig. 3.1. Complications of Nastaleeq writing style

languages (weber[6]) of world. The number of research work is ongoing to preserve its literature. Urdu is an Arabic script language, mostly written in Nastaleeq writing style. Nastaleeq writing style is also followed in many other languages including Persian, Pashto, Panjabi, Baluchi, and Siraiki [13, 14]. Successful OCR for Nastaleeq text therefore becomes imperative.

Generally, Urdu language OCR systems can be divided into three categories based upon text's segmentation, which includes character based [11, 21, 40-43, 45, 46, 61], ligature based [15, 24, 31, 59] and sentence based systems [48, 50]. Currently, researchers are focusing more on ligature and sentence based OCRs. All these OCR systems mainly rely on of document images' segmentation into text lines, ligatures and characters. Complexities of Urdu Nastaleeq writing style are the major obstacles for its segmentation into lines and ligatures. Major complexities of Nastaleeq writing style include context sensitiveness, compactness, overlapping, cursive nature, diagonality, and many others [14] as presented in Fig. 3.1.

Nastaleeq accommodates more characters in less space because of its diagonality and cursive nature [14]. Characters are stacked from right to left and top to bottom into a word. For instance, the starting character of each ligature in Fig. 3.2 is same but different in shape [21]. In case of Urdu Nastaleeq writing style, the number of possible shapes of certain characters reach up to 60 [12, 62]. Therefore, it is very difficult to segment text in Nastaleeq writing style due to large number of character shapes, inter-ligature and intra-ligature



overlaps, and context sensitivity of text [14]. Furthermore, in Nastaleeq writing style, there

[6] http://www2.ignatius.edu/faculty/turner/languages.htm (accessed 10/05/2016, 4:25am)

is no consistency in spacing among the words, ligatures and sentences. Some ligatures of sentences intrude the boundaries of other sentences that are written above or below lines, as presented in Fig. 3.3.

In the first line of the paragraph, the character "م" is connected with the word "بجلی" of second line. Besides, ligature "کے" of third line is entering in the boundary of second line. Also, ligature "پر" of third line is crossing the border of fourth line. Thus, in Urdu Nastaleeq text, there exists no fixed inter ligature, inter words and inter sentences gap as demonstrated in Fig. 3.3.

3.2 Related work

An overview of contributions in the area of Urdu line and ligature segmentation is briefed in the following subsections.

3.2.1 Line segmentation

Text lines extraction is a very crucial step in segmenting procedure. The algorithms applied for Urdu page segmentation into lines include horizontal and vertical projection, x-y cut, smearing, geometric, ridge based, and zone-based horizontal projection algorithms.

The pioneer work [31] on Urdu OCR used horizontal projection profile method for text lines extraction, followed by component labeling and vertical projection profile methods for character extraction. Horizontal and vertical projection methods were applied on scanned document image for text lines and character segmentation, respectively [54, 57]. Although, horizontal projection segmented the text lines correctly but it could not segment lines with merged ligatures.

Recursive XY cut, whitespace analysis, constrained text-line detection, docstrum,

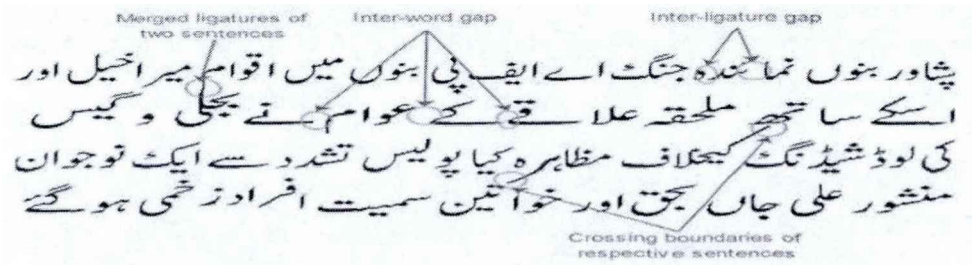
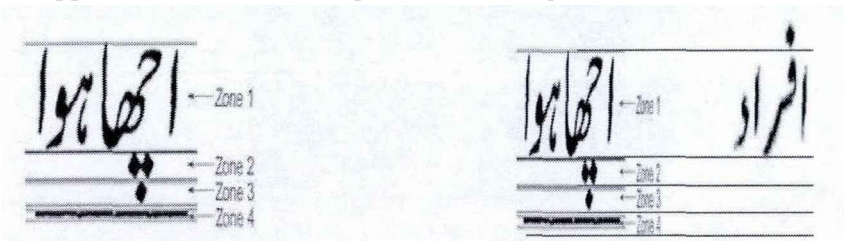


Fig. 3.3. A paragraph of Nastaleeq writing style and its complexities

voronoi-diagram based algorithm, and smearing algorithm were used [101] on five scripts including Nastaleeq were segmented into text lines, and the highest accuracy reported for Urdu was 65.4%. The results are not promising for Urdu text, because of its overlapping nature, non-equispaced lines and non-availability of clear demarcation.

Geometric algorithms for layout [102] were adopted and tailored for getting the correct layout of Urdu Nastaleeq documents [103]. The authors achieved accuracy of text line detection ranging from 72% to 93% for different genres of document images. Ridge based approach [104, 105, 106, 107] was applied for text line extraction, while the method in [28] was adopted for reading order. The achieved accuracies for Arabic and Urdu document images were 96% and 92%, respectively. Although these methods are accurate but lack proper allocation of misplaced diacritics/dots.

Zone based global horizontal projection method [109] was proposed for segmenting Nastaleeq's Urdu text lines. The method segmented document images containing touching ligatures across lines with the accuracy of 99.11%. In this method, there was no zone mentioned above the zone 1, as depicted in Fig. 3.4(a). Therefore, it overlooks the diacritics of the text line that appear above zone 1, as presented in Fig. 3.4 (b).



a) Zone based line segmentation[109] b) Diacritics/dots above zone 1

Fig. 3.4. Zone based Urdu Nastaleeq text line segmentation

In Urdu Nastaleeq, it is possible to have some diacritics/dots above the zone 1(i.e.; main text). So, there are possibilities of assigning diacritics/dots inaccurately, as depicted in Fig. 3.4(b). Implicitly, three zones were suggested in [108] instead of 4 zones methodology for page segmentation as shown in Fig. 3.5.

This strategy takes into consideration the dots/diacritics even above zone 1. A recent study [9] also used zone based method [109] with dilating document images before segmentation for getting more correct size of zones, which resulted in 98.7% line segmentation accuracy. This method confronts difficulties for segmenting merged text lines.

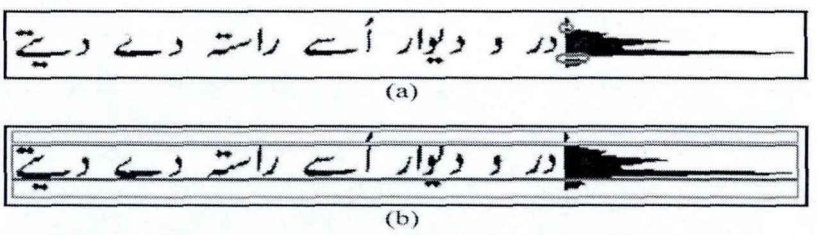


Fig. 3.5. Zone above zone 1 [108]

3.2.2 Ligature Segmentation

A ligature is composed of one or more characters joined together [109], and the combination of one or more ligatures form words in Arabic script (Like Urdu language). A ligature is comprised of two parts; the primary component is made up of main stroke, while

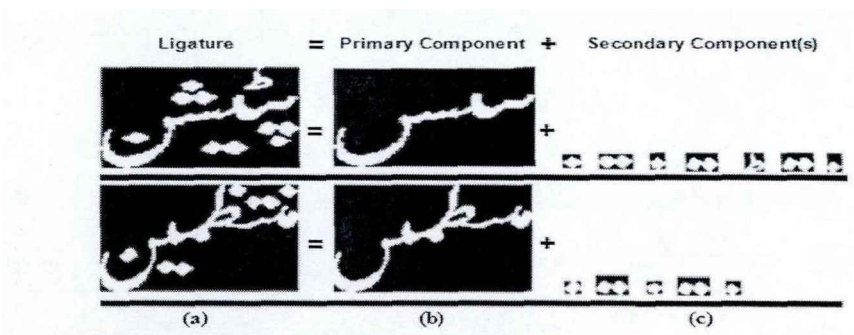


Fig. 3.6. Primary and secondary components of two ligatures

diacritics are nominated as secondary strokes as presented in Fig. 3.6.

Considerable efforts were devoted for line segmentation into ligatures like vertical histogram method [48] and [102], horizontal projection method [21], and bounding box method [15, 59, 103, 109, 110].

Individual ligatures extracted by vertical histogram of text line in Naskh was described in [48] and [102], but this method is not applicable for Nastaleeq because of inter-ligature overlapping. In another work [21], vertical projection was applied for getting ligatures and character on extracted lines by horizontal projection method. Ligatures were also segmented by connected component method supplemented with baseline and centroid information, while achieving 94% segmentation accuracy [15]. Connected components were extracted by bounding box methods from text lines [59] and [103], and then secondary components were allocated to primary components based on the detected baseline. Text lines were segmented into connected components that were categorized into 6 types [109], and then secondary components were merged together with primary components to get the resultant ligatures based upon certain heuristics, thus, achieving 99.02% accuracy. Another study [110] first segmented lines into connected components, and then allocated secondary components to primary components with 92.5% accuracy.

3.2.3 Motivation

The most recent line and ligature segmenting algorithms [109, 110] proposed the use of 4 zones for line segmentation, and 6 categories of connected components for constructing a final ligature. The authors did not utilize the baseline information that is proved to be very much decisive in proposed line segmentation algorithm. Also, ligature segmentation [109] relies on 6 heuristics, among which only one heuristic uses baseline information. Relying more on zonal and heuristics information for line and ligature segmentation becomes very much context and font dependent. Lines were segmented with perfection [103, 104], but misplaced dots/diacritics were not allocated to their respective lines. Thus, here motivation is to reduce the complex mechanism of zoning and their merging decisions for line segmentation. The proposed approach mainly relies on baseline information with three zonal method for dots/diacritics allocation, and quantitative values (width, height, centroids, and overlapping) for ligature segmentation.

3.2.4 Contribution

This chapter presents two algorithms for line and ligature segmentation. These algorithms can also be applied to other Nastaleeq based scripted languages, such as Arabic, Persian, Pashto, and Sindhi etc. Specifically, the proposed line segmentation algorithm depends solely on horizontal projection, split positions (proposed in this chapter and named as Curved-Line-Split (CLS) algorithm) and baseline information. In proposed approach, the allocation of dots/diacritics to proper text line is also properly addressed by employing three-zonal method. For ligature segmentation, the proposed algorithm employs width, height, centroids, and overlapping quantitative information as opposed to heuristics [109, 110]. The accuracies achieved by both the proposed line and ligature segmentation algorithms are 99.17% and 99.08%, respectively, which are better than the previous contributions [109, 110, 104].

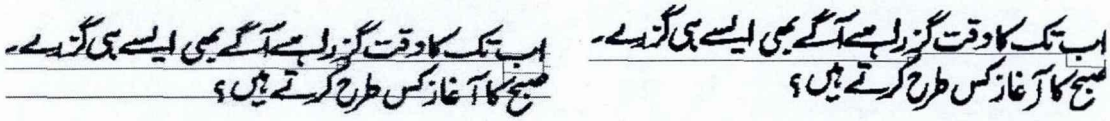
3.3 Line Segmentation

This section details about segmentation algorithm of a page into constituents text lines, which employs global horizontal projection method augmented with the proposed CLS algorithm. Additionally, it contains steps for conflicts resolution regarding components crossing split lines based upon baseline information.

3.3.1 Curved Line Split Algorithm

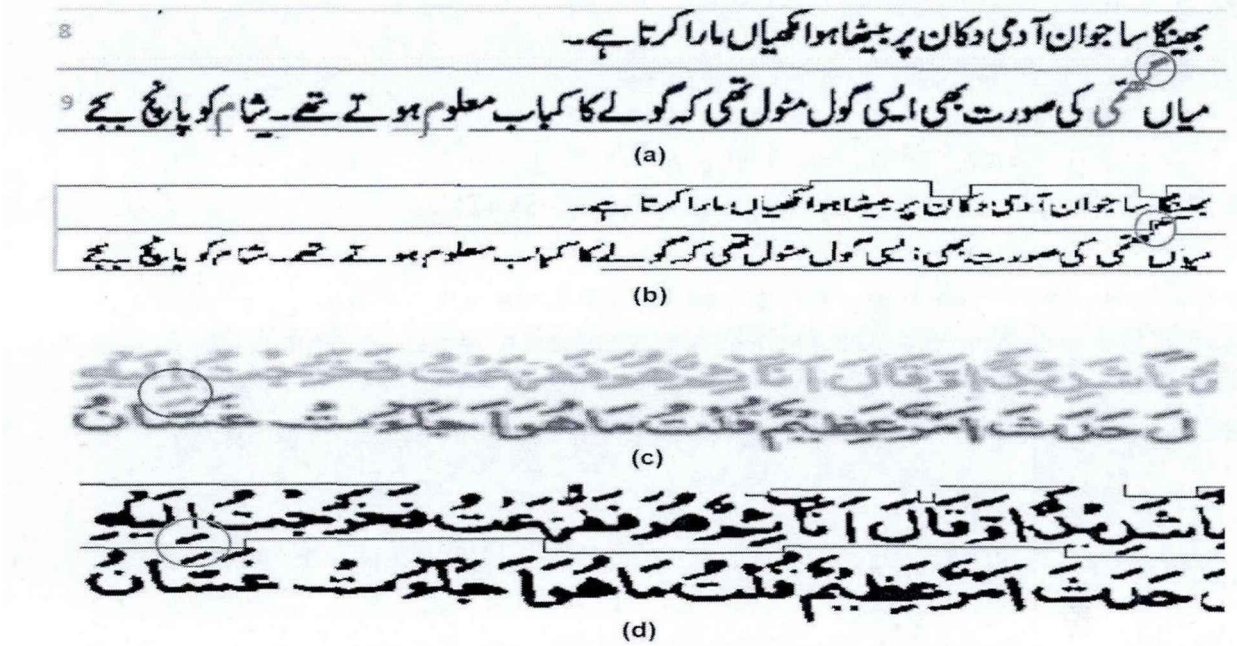
The proposed CLS algorithm can even find out a curved (not straight) demarcation line between every two consecutive text lines, when there is no straight demarcation line. The basic idea of CLS algorithm is as follows:

Basic Idea: Algorithm starts a demarcation line from middle of baselines of consecutive text lines. It traverses a path from left to right on horizontal path, if it finds text pixels it moves up or down vertically until it finds a text pixel or baseline, and then from half of the vertical movement it starts again horizontal traversal in between two lines. The procedure of CLS is presented in Figs. 3.7 and 3.8. The CLS algorithm detail is presented in Algorithm 1.



a) Working of CLS algorithm b) Demarcation line, output of CLS algo

Fig. 3.7 Application of CLS algorithm and its output with demarcation line



(a)

(b)

(c)

(d)

Fig. 3.8. Improper segmentations encircled red in (a)[109],(c)[104] are solved by the proposed Curved Line Split algorithm encircled green in (b) and (d).

The detail of CLS algorithm is presented in Algorithm 1.

Algorithm 1: Curved Line Split

Data: Document Image Page, initial split row numbers
Result: coordinates for demarcation/split lines
begin
  Draw base lines for every text line. Base line will be the maximum intensity *pixels'row* in between every two consecutive initial split row numbers.
  for *each initial split row number* **do**
    x= initial split row number, y=1, set *resolve*=0, *stuck*=0.
    Store (x,y) as starting coordinates for demarcation line.
    x = initial split n^{th} row number, y=1 .
    Direction to move up or down when find an obstacle=*d*=down.
    *bl*1= Base line of current line , *bl*2 = Base line of previous line.
    while y <= *Pages Max column and* $bl1 < x < bl2$ **do**
      Move forward on x^{th} row by increasing y until find an obstacle(text pixels).
      Store $(x, y-1)$ as coordinates for demarcation line.
      Move in d direction through column $(y-1)$ until find an obstacle (text pixels).
      d=opposite of previous direction.
      Store $(x/2, y-1)$ as coordinates for demarcation line.
      if $pre_y = y-1$ **then**
        stuck= stuck+1
      $pre_y = y-1$
      if *stuck(No change in value of y) for 4 times and resolve* < *4* **then**
        go back 3 pair values in coordinates for demarcation line
        if *current rowsx > next rows x in coordinates for demarcation line* **then**
          set d=down
        else
          set d=up
        set $x = x$ of previous row coordinates for demarcation line.
        set $y = y$ of previous col coordinates for demarcation line.
        $pre_y = y$
        set $resolve = resolve + 1$ and $stuck = 0$.
      if *stuck(no change in value of y)=4 and resolve=4* **then**
        Move through info pixels on x row and increase y until information pixels are over.
        Set $resolve = 0$ and $stuck = 0, pre_y = y-1$.
      $y = y+1$.
  Return coordinates for demarcation/split lines

3.3.2 Line Segmentation Algorithm

Line segmentation algorithm is applied on Urdu Nastaleeq document image for getting segmented text lines. Classical horizontal projection based segmentation method is augmented with a proposed CLS algorithm. This augmentation successfully overcomes the problems, such as text line split position, overlapping, merged ligatures and ligatures crossing line split positions. Horizontal line segmentation algorithms [21, 108, 109, 110] can split two text lines if there is a clear horizontal demarcation. The CLS algorithm overcomes this flaw and can find a demarcation boundary in case of non-equispaced text lines and non-availability of clear demarcation. The main steps are described as follows:

1. Find Line-Peaks: In this step, an image of Urdu Nastaleeq script I_U is converted into binary image I_{UB} by global thresholding method. This binarized image is inverted for getting an image $I'_{H\times W}$ with white text on black background. Then, Horizontal Projection (HP) is calculated for the entire document.

$$\text{HP(x)} = \sum_{y=1}^{W} I'(x,y), \forall x \in [1,H] \tag{3.1}$$

Find all peaks and troughs of horizontal projection.

$$\text{Peaks: } \forall x \; find \; x_i \; s.t. \; x_{i-1} < x_i > x_{i+1} \tag{3.2}$$

$$\text{Troughs : } \forall x \; find \; x'_i \; s.t. \; x'_{i-1} > x'_i < x'_{i+1} \tag{3.3}$$

First of all, peaks are scanned to find out all local highest valued peaks. Generally, such peaks start with a lowest peak value after a trough followed by step-wise higher valued peaks until one gets a local highest valued peak. Then, the values of next coming peaks gradually decrease until it reaches another trough. Such local highest valued peaks are termed as Line-Peaks, which are accompanied by gradual lower valued peaks on both sides representing one line. Line-Peaks represent center/base with maximum pixel intensity of text lines in a page image.

$$\text{Line} - \text{Peaks: } \forall x \; find \; all \; x''_c \; s.t$$

$$x_{i-1} < x''_c > x_{i+1} \tag{3.4}$$

2. Next, the average number of pixels' rows per text line.

$$\text{Average Text Line(TL) Height: } Avg_H(TL) = \frac{|x|}{|x''|},$$

Where, $Avg_H(TL)$ *is* Average Text Line(TL)Height,

$|x|$ *represents total rows of pixels,*

x'' *represents number of Line_Peaks* , (3.5)

3.False text lines: Once again scan the peaks, if lowest troughs of some Line-Peak come before (¾)th of the average text line height, then, that line is considered as false Line-Peak and excluded from total Line-Peaks.

$\forall x''_c$ *find*: $a = |x''_{ci-1} - x''_{ci-2}|$,

b= $|x''_{ci} - x''_{ci-1}|$,

c= $|x''_{ci+1} - x''_{ci}|$,

$$if\ c < \frac{3}{4} Avg_H(TL)\ \wedge\ a < c, \qquad a = a + b,$$

$$Eliminate\ x''_{ci}\ from\ \text{Line} - \text{Peaks} \tag{3.6}$$

4.Missed text lines: All Line-Peak values must be greater than the highest trough value; otherwise the text lines with Line-Peaks less than the highest trough will be missed. After conforming step 3, the value of all such Line-Peaks hypothetically increased more than the highest trough value to fix the problem of missed lines.

Highest Trough : $max\ x'_T = H_T \forall x\ find\ x'_i\ s.t$

$$x'_{i-1} > x'_i < x'_{i+1}\ ] \tag{3.7}$$

Missed Text Lines : $\forall x''_{ci} < H_T$,

$$x''_{ci} = x''_{ci} + (|H_T - x''_{ci}|) \tag{3.8}$$

5.Segment page image into text lines: Pass initial split row numbers to CLS algorithm. The initial candidate split row numbers are the lowest trough values between every two Line-Peaks. Convert the text page into connected components format, so that every component is represented by a different number. Then, split underlying page according to the coordinates for demarcation of lines returned by CLS algorithm.

6. Assignment of components: Assignment of misplaced and crossing components to proper text line can easily be done with the information of baseline of every text line.

Components that are crossing the demarcation line are referred as crossing components. For visualization of this concept, four cases are briefed as follows:

Case 1: Crossing component touching one baseline: Move the crossing component to the text line where it is touching the baseline.

Case 2: Crossing component not touching any baseline:

a) Here, find the top row, bottom row, left most column and right most column numbers for under consideration crossing component.

b) Next, start search by moving above from the top row and below from the bottom row while checking every pixel from left most column to right most column for finding the nearest text pixel of any other component.

c) Move this crossing component to the corresponding side where text pixel is found earlier. In case of tie, move component to the side where search found component is touching baseline. If both are not touching the baseline, then perform step a, b and c for both newly found components, but search is performed in the same direction where these new components are found.

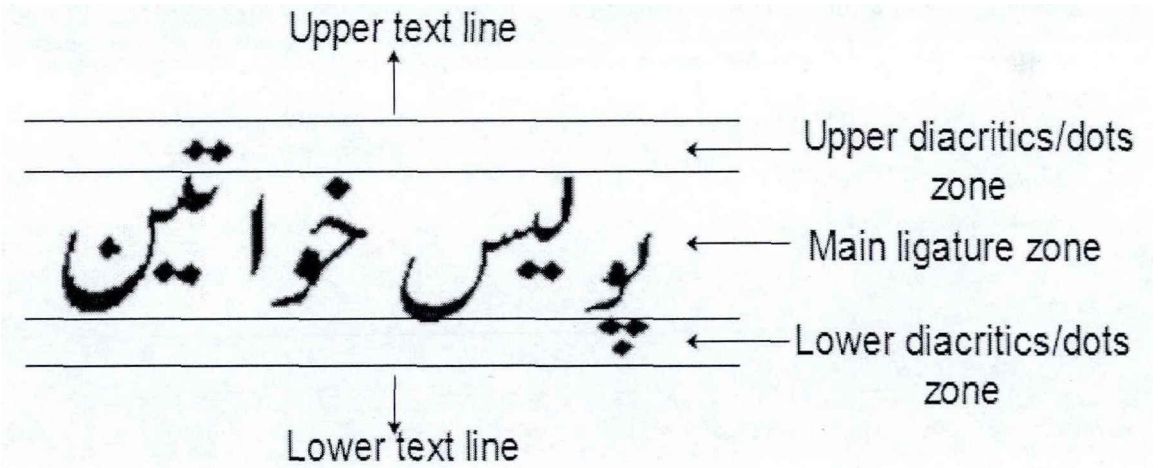


Fig. 3.9. Upper and lower diacritics/dots zones scanning

Case 3: Connected component touching neither base lines nor split line: Some diacritics/dots might neither crossing any base nor split line. Such components might be segmented in wrong text line. Therefore, scan every text line for any text component

measuring approximately (¼)th height of each line. These areas are referred as upper and lower diacritics/dots zones as presented in Fig. 3.9.

There may be possibilities of attaching diacritics to wrong lines, most probably in upper and lower diacritics/dots zones of every text line except first and last ones. First text line may only have lower diacritics/dots zone, while last one may have only upper diacritics/dots zone.

In these portions, if some component is found then it is moved to the line where it has minimum distance from the neighboring component. This step can easily be performed as follows:

a) For every component found in upper or lower diacritics/dots zone, where:
   i. The component must not touch baseline.
   ii. The component must not be greater in size than its respective zone.
b) Find the top and bottom row of each component. Also, find the left and right most column numbers of its maximum span/width.
c) Start searching above and below from the component's top coordinates and bottom coordinates, respectively, for finding the nearest text pixel.
d) Move this component to line where text pixel is found earlier. In case of tie, move the component to the line where search finds the component that is touching baseline. If both are not touching baseline then for breaking the tie perform step a, b and c for both newly found components but search is performed in the same direction where these new components were found.

Examples: Each step of the propose line segmentation algorithm is explained by applying it on a complex text page presented in [103] as shown in Fig. 10(a). Also, algorithm is applied on other sample pages in earlier works [109, 110, 104] are shown in Fig 10 (b), (c) and (d), respectively.

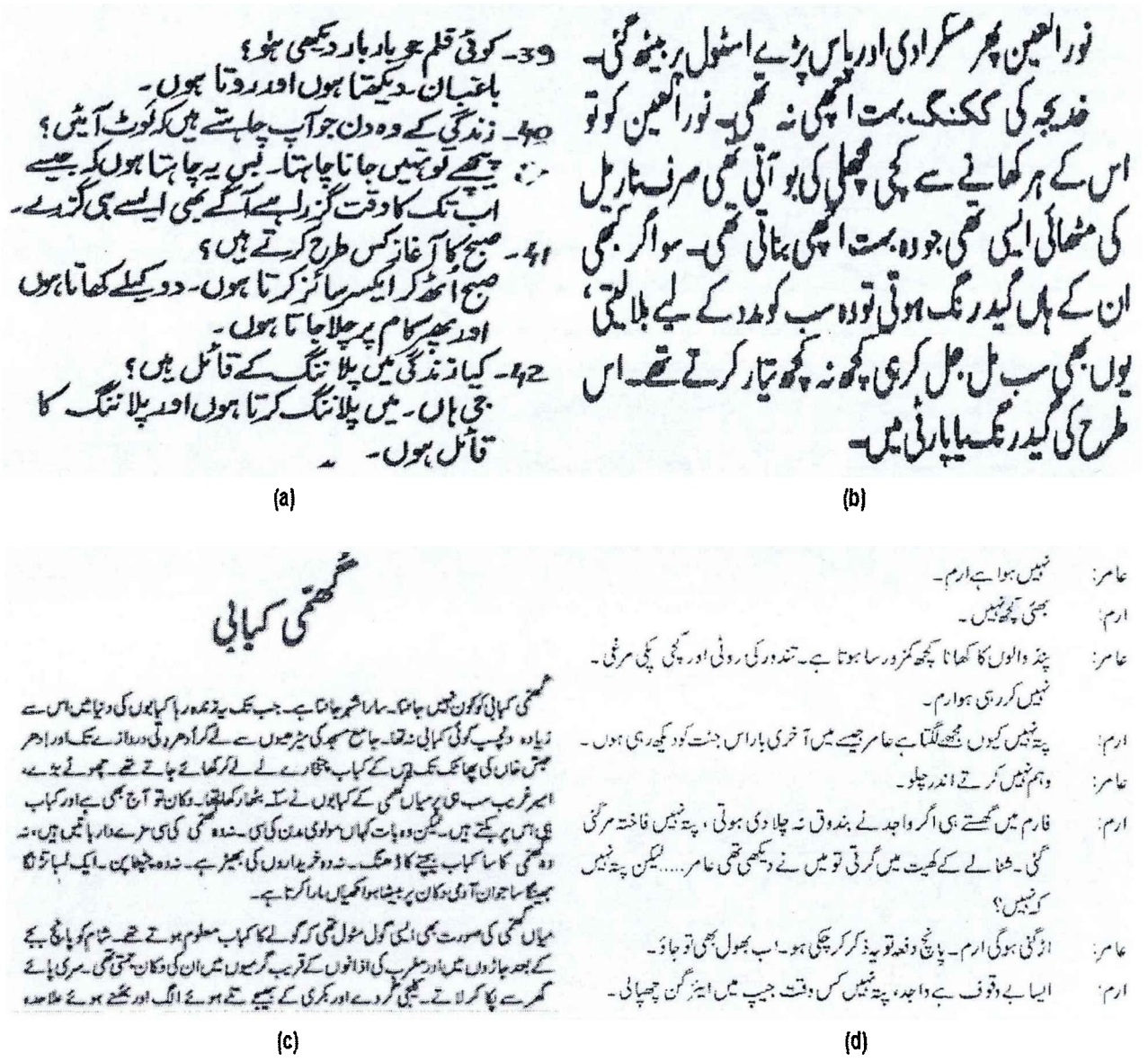
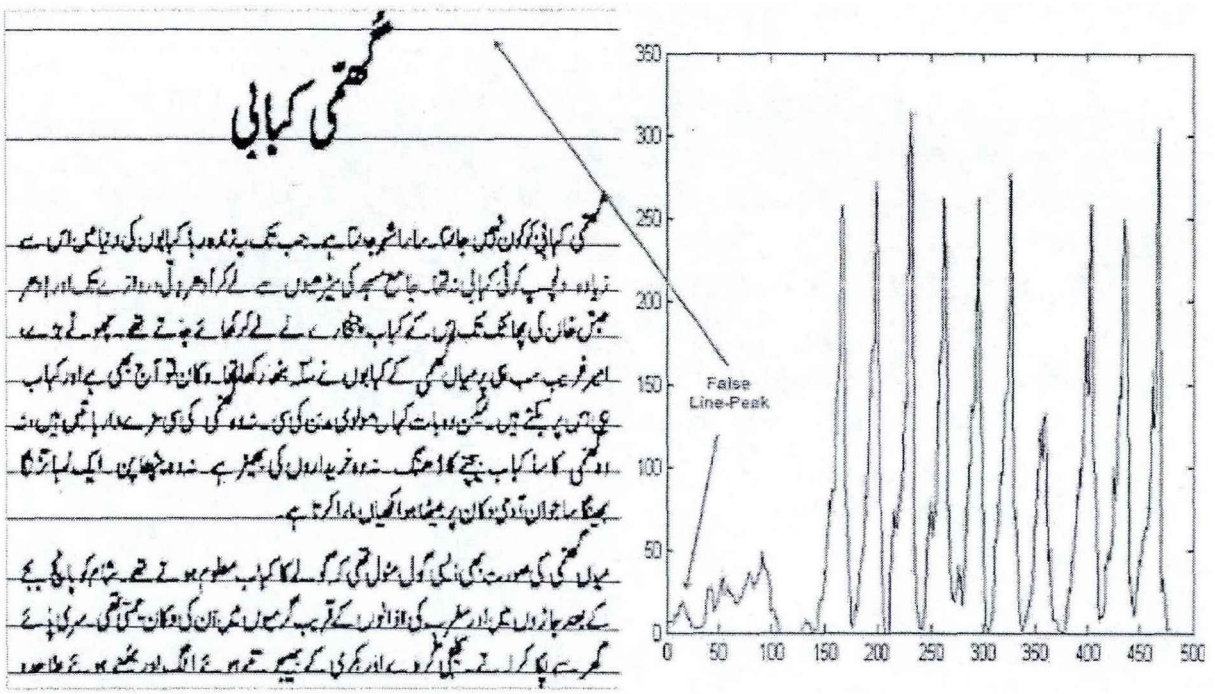


Fig. 3.10 Sample Urdu Nastaleeq pages (a) from [103], (b) from [104], (c) from [109], (d) from [110]

Steps 1-3: Top most Line-Peak/basline in Fig. 11(a) represents false Line-Peak that may result a false text line. As, Fig. 11(b) represents that this Line-Peak and its corresponding troughs last well before (¾)th of average text line height, so it is eliminated from Line-Peaks. Furthermore, this eliminated Line-Peak is merged with the smaller height neighbored line.



(a) (b)

Fig. 3.11 False Line-Peak represents false line

Case 4: Crossing component touching both baselines: This situation comes along when two ligatures are connected mistakenly. The position of the joint of two ligatures can be found with minimum number of pixel. Start from the split position of text lines along the crossing component above and below, and search for minimum pixel connectivity nearby split position. This search must not move more than one fourth of the height of a text line. Otherwise, split this component from the text line split position.

Step 4: Fig. 3.12 depicts that all Line-Peaks less than the highest trough will be missed, as last line of the page is a missed line.

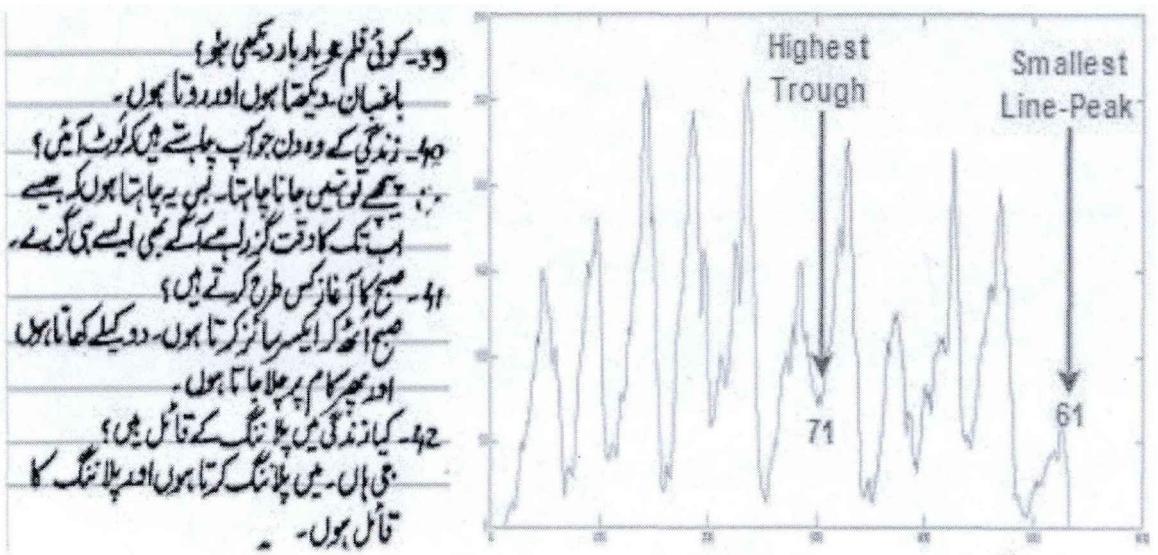
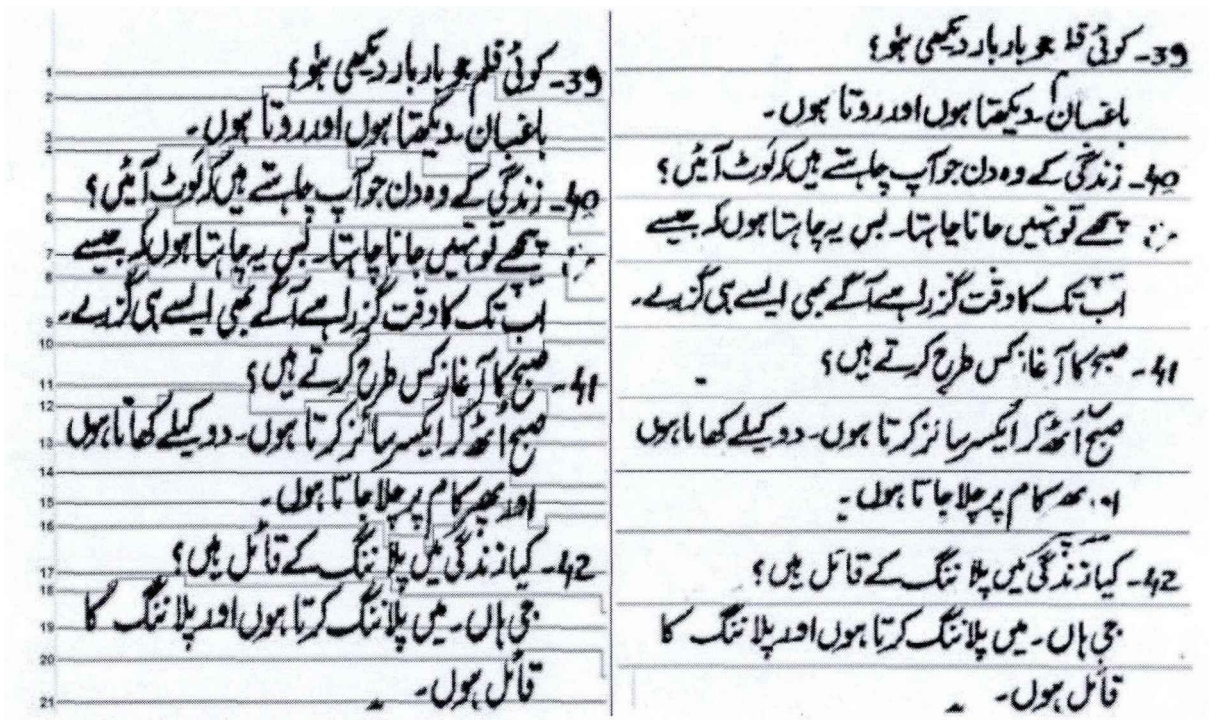


Fig. 3.12 Last line represents missed Line-Peak as its value 61 is less than highest trough value 71

Step 5: Initial lines split after demarcation by CLS algorithm as shown in Fig. 3.13.



(a) (b)

Fig. 3.13 Steps 5, (a) odd and even numbered lines are representing base and split lines, respectively. Initial segmented text lines in (b).

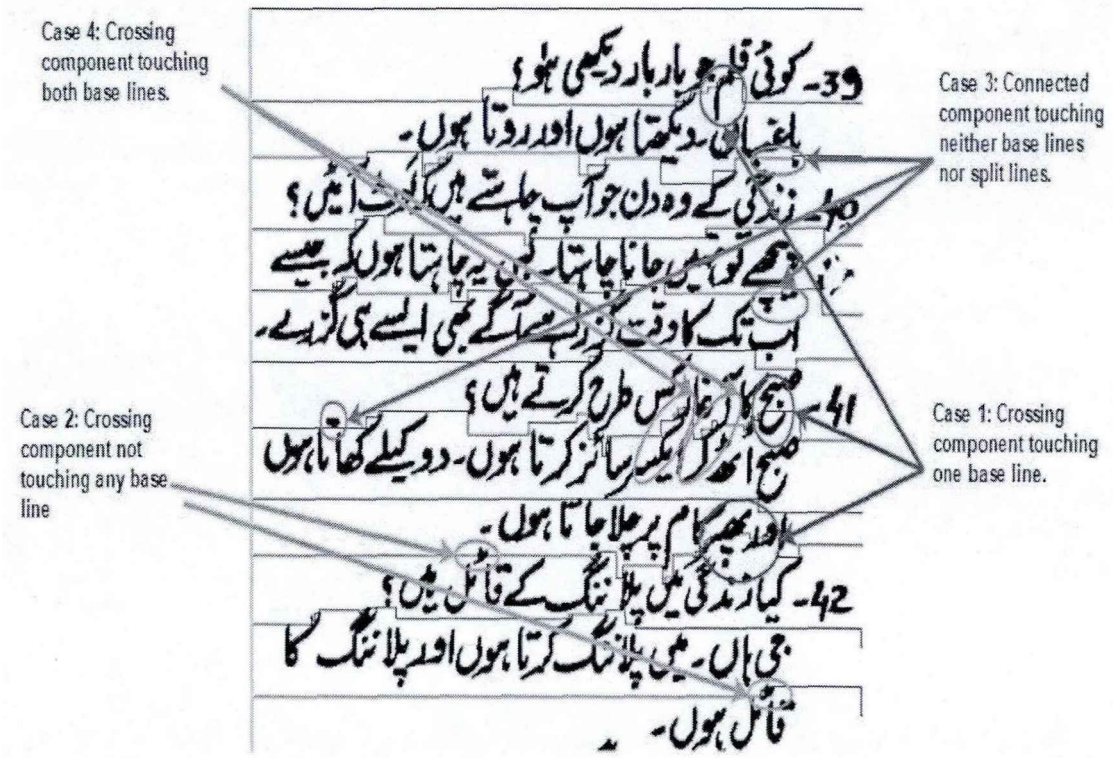


Fig. 3.14 Four cases for resolution of conflicting components

Step 6: Resolution of 4 cases is shown in Fig. 3.14- 3.16:

Fig. 3.15. Working of the proposed line segmentation algorithm. Red encircled components of previous step are resolved in next step, encircled green, (a) to (b) crossing component touching one baseline, (b) to (c) crossing component does not touching any baseline, (c) to (d) component neither touching baselines nor split

3.4 Ligature segmentation

Ligature segmentation algorithm splits text lines into constituent ligatures. A ligature is usually composed of 1 to 8 characters forming one primary and zero or more secondary connected components. Main body of a ligature is known as primary component that is made up of longest primary stroke. Secondary components are usually dots and diacritics, as presented in Fig. 3.6. The basic idea of proposed ligature segmentation algorithm is to find all components and then allocate secondary components to proper primary component as illustrated in Algorithm 2.

The proposed line segmentation algorithm extracted all constituent text lines with only two diacritics out of place encircled red as indicated in Fig. 3.17.

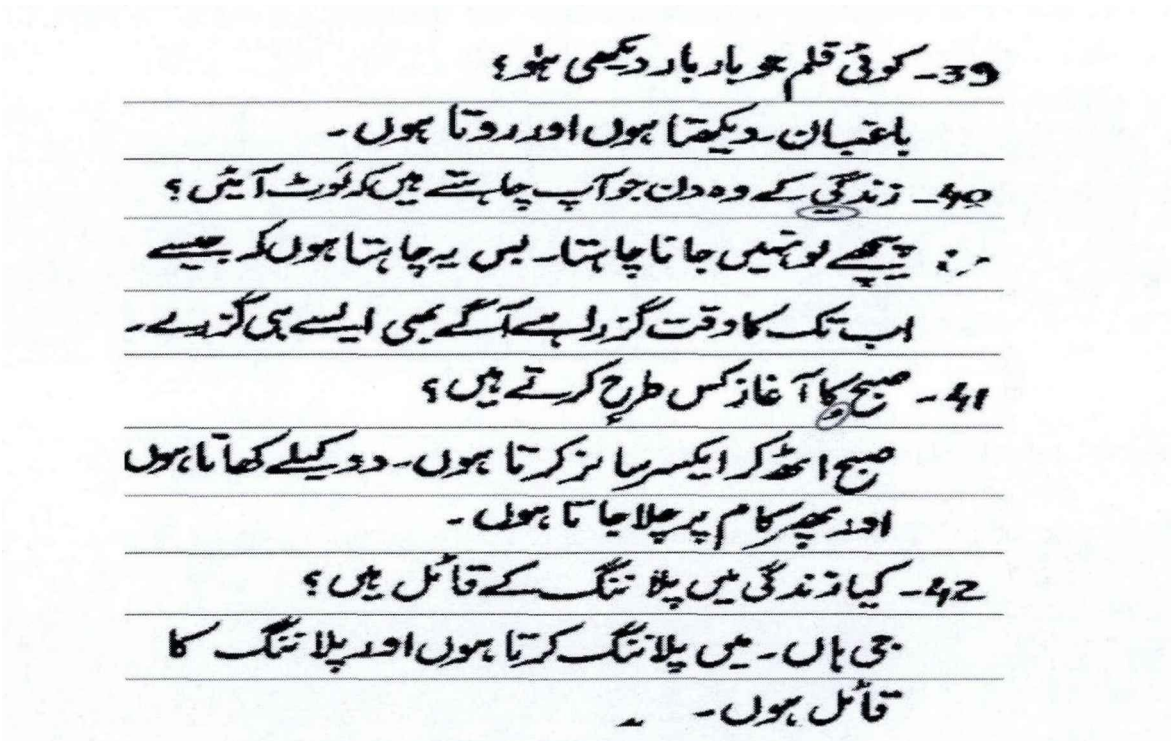


Fig. 3.17 Segmented text lines by the proposed line algorithm

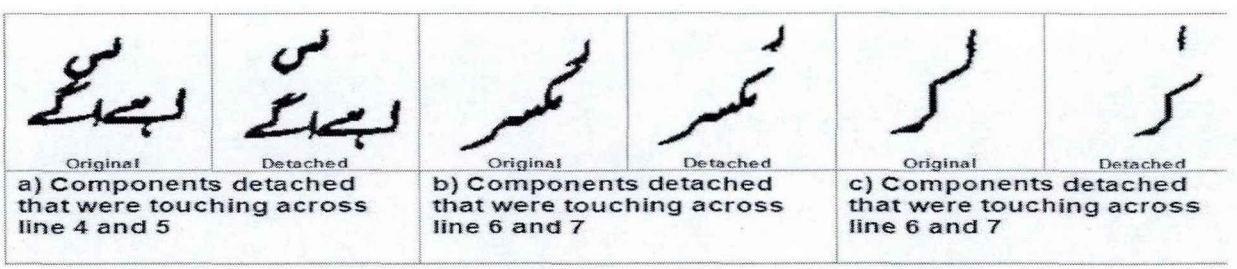


Fig. 3.16 Example of crossing components, touching both base lines

Algorithm 2: Ligature Segmentation Algorithm

Data: S = Urdu Sentense Images
Result: List of ligature images for a given Urdu sentence image S_k , $L^k = \{Ligature_i^k\}_{i=1}^m$
begin
  for k= *1 to* S **do**
    T^k = Extract information(connected comp(S_k))
    T^k = Decide primary secondary components(T^k)
    L^k = Allocate secondary to primary(T^k)
  return $L^k = \{Ligature_i^k\}_{i=1}^m$, m=number of ligatures of S_k

The proposed ligature segmentation algorithm therefore has three steps: 1) The sentence images are split into connected components and information is extracted. 2)Connected components are then categorized into primary and secondary classes by examining width, height, coordinates, overlapping and centroids along with the baseline information of the sentence. 3) Secondary components are associated to proper primary components to form ligatures. The detailed explanation of these 3 steps is provided as follows:

3.4.1 Extract Information

Algorithm 3 finds connected components and collects overlapping, height, width, centroids and baseline information.

Algorithm 3: Extract information

Data: S_k = Urdu sentence image,
Connected Component=CC
Result: Table T^k,containing information regarding connected components of a sentence S_k
begin
  CC = Find connected components forS_k
  for *each connected components* CC_i **do**
    Find starting and ending columns number
    Find starting and ending rows number
    Touching baseline or not
    Calculate height and width
    Overlapping connected component
    Centroid
    Save all above information about CC_i in T_i^k
  return (T^k)

3.4.2 Decide Primary and Secondary Components

This step classifies the connected components into primary and secondary on the basis of collected information in previous step. It is depicted in Algorithm 4.

Algorithm 4: Decide primary and secondary components

Data: T^k = containing information of connected components, Connected Component=CC, Secondary Component Height < Connected Comp Height $CCH = \frac{1}{2}$ of the height of a text line

Result: Updated Table T^k =, containing information regarding connected components and their types.

begin

- **for** *each* CC_i *that overlaps with any other* CC_j *in* T^k **do**
  - **if** CC_i *is not diacritic and* SCC_j *is a diacritic* **then**
    - **if** CC_j *already allocated to some primary* CC_o **then**
      - **if** CC_i *completely overlaps* CC_j *OR centroids of* CC_i *and* CC_j *are closer than* CC_o *and* CC_j *OR overlapping area of* CC_i *and* CC_j *is more than* CC_o *and* CC_j **then**
        - update CC_j as secondary component of CC_i in table T^k
      - **else**
        - **if** CC_j *already not allocated to some primary component* **then**
          - **if** CC_i *partially overlaps* CC_j *OR height of* $CC_j << CCH$ **then**
            - update CC_j as secondary component of CC_i in table T^k
    - **else**
      - **if** *height of* $CC_j < CCH$ **then**
        - update CC_j as secondary component of CC_i in table T^k
    - **else**
      - **if** *height of* $CC_i > CCH$ *and height of*$CC_j < CCH$ *and* CC_j *is touching baseline* **then**
        - update CC_j as a secondary component of CC_i in table T^k
- return T^k

3.4.3 Allocate Secondary to Primary Components

In Algorithm 5, pixels of secondary components are moved to their respective primary. Firstly, all the non-conflicting (not touching baseline) secondary components are moved to respective primary components. The conflicting secondary components (touching the baseline) are decided based on their size and height. If size/height is less than the threshold size/height of secondary components, then these components are declared as secondary and copied to the respective primary components, otherwise, these components are declared as primary. These steps are depicted in Algorithm 5.

The proposed ligature segmentation algorithm is applied to the sentences of UPTI data set as well as to the segmented sentences presented in Section 2. For example, three sentences and corresponding ligatures are depicted in Fig. 3.18.

Algorithm 5: Allocate secondary to primary components

Data: T^k = contains information of connected components and their types, Secondary Connected Component=SCC, Primary Connected Component=PCC, Secondary Component Height < Connected Comp Height $CCH = \frac{1}{2}$ of the height of a text line
Result: List of ligatures $L^k = \{Ligature_i^k\}_{i=1}^m$
begin
  for *each* SCC_i *of* T^k **do**
    if SCC_i *is not touching baseline (Non conflicting* SCC_i*)* **then**
      Copy pixels of SCC_i at proper coordinates on its allotted PCC
    else
      if SCC_i *touching baseline and height of* $SCC_i < CCH$ **then**
        Declare SCC_i as not touching baseline in T^k
        Copy SCC_i at proper coordinates to its allotted PCC
      else
        Declare SCC_i as PCC in T^k
  for *each* PCC_i *of* T^k **do**
    Save PCC_i as a $\{Ligature_i^k\}$, i^{th} ligature of k^{th} sentence
  return $L^k = \{Ligature_i^k\}_{i=1}^m$, m=number of ligatures of S_k

3.5 Results and analysis

In this section, the results of the proposed line and ligature segmentation algorithms are discussed. Results are compared with the prevalent studies.

3.5.1 Results of the Line Segmentation Algorithm

The proposed line segmentation algorithm is tested on 47 pages. Lines are segmented 100% for 7 pages taken from earlier papers [103, 104, 109, 110]. Remaining 40 pages are taken from single columned editorials of an online Urdu newspaper.

The proposed line segmentation algorithm segmented lines of 47 pages with the accuracy of 99.17% that is better than accuracies reported in prevalent work. Overall accuracy of the proposed line segmentation algorithm is 99.17% as presented in Table 3.1.

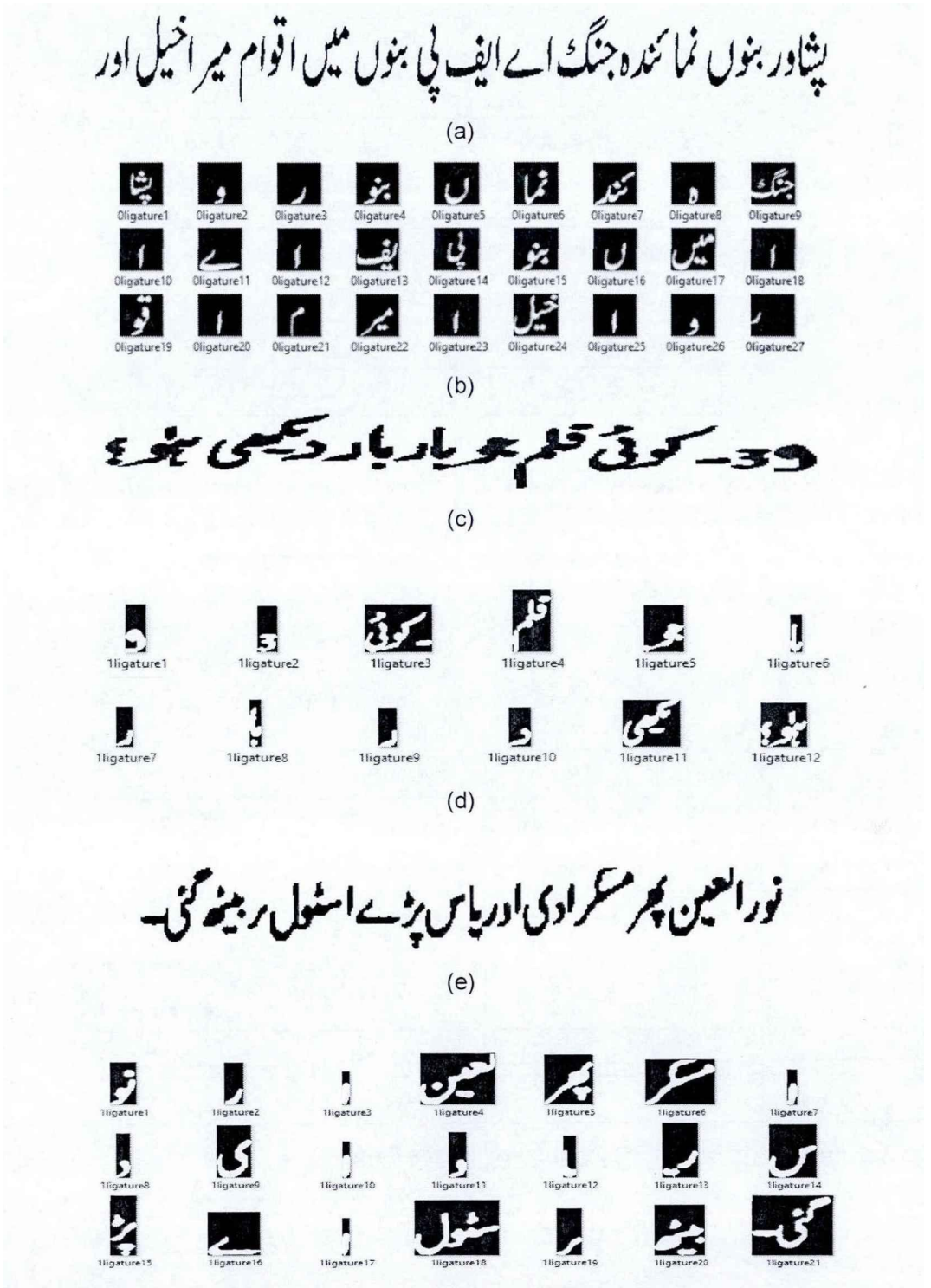


Fig. 3.18 Total 27, 12 and 21resultant ligatures of above displayed sentences are taken from UPTI data set, [103] and [104], respectively. First sentence of UPTI data set and its segmented ligatures are represented in (a) and (b). A sentence of a text page given in [103] and its segmented ligatures are represented in (c) and (d), respectively. Similarly, a sentence of a text page given in [104] and its segmented ligatures are represented in (e) and (f), respectively.

Table 3.1

Results of the Proposed Line Segmentation Algorithm

Source	Script	Pages	Lines	Correctly segmented lines	Accuracy
[109]	Urdu	1	11	11	100%
[110]	Urdu	2	28	28	100%
[103]	Urdu	1	11	11	100%
[104]	Urdu	2	16	16	100%
[104]	Arabic	1	9	9	100%
Newspaper	Urdu	40	532	527	99.06%
Average accuracy					99.17%

Table 3.2

Comparison Line Segmentation Algorithm

Source	Pages	Lines	Detected lines	Accuracy
[109]	448	Not reported	Not reported	99.11%
[110]	30	310	306	98.70%
[103]	25	Not reported	Not reported	Books and magazines= 90% Digest=80% Newspaper= 72%
[104]	20	Not reported	Not reported	92%
This study	47	607	602	99.17%

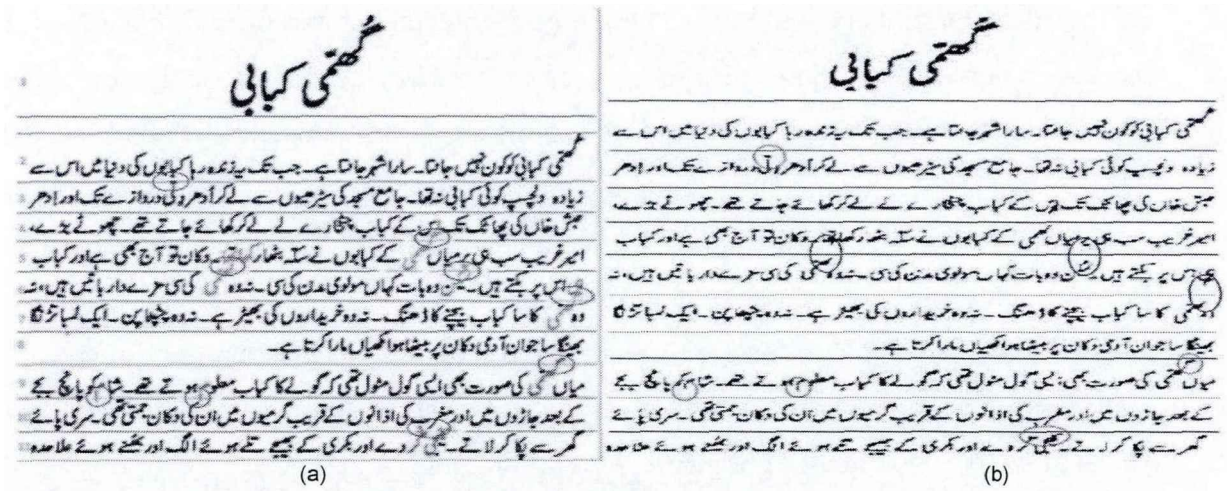
3.5.2 Comparison of Line Segmentation Algorithms

Prevalent contributions in the area Urdu line segmentation used their own data sets, as there is no publically available data set. Therefore, accuracy depends upon the complexity of text images used for segmentation. One can observe very complex text images shown by [103] and [104] in their contributions and highest reported accuracy is 92%, as depicted in Table 3.2. The authors proposed multicolumn Urdu text line detection mechanism. However, they have not incorporated mechanism for dots/diacritics allocation. This work selected 4 text page images from their work and the proposed algorithm performed better with an accuracy of 99.17% because of diacritics allocation mechanism. Lehal [109] and Din et al. [110] reported 99.11% and 98.70% accuracies.

Line segmentation by [109] after segmentation of merged lines

Initial line segmentation by the proposed line segmentation algorithm

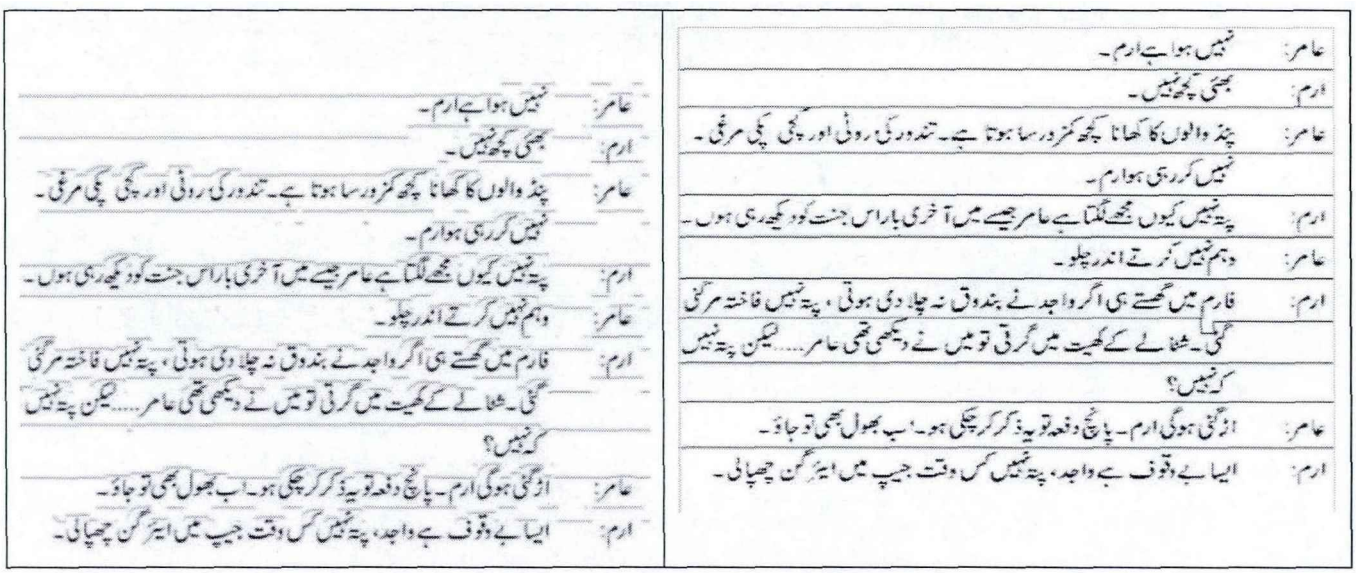
Fig. 3.19 Initial line segmentation by [109] and the proposed line segmentation algorithm



(a) (b)

Fig. 3.20 Final segmented text lines, (a) by [109] and (b) by the proposed line segmentation algorithm

Line segmentation [8] draws a straight boundary line between two text lines that may not always be straight. Therefore, it needs to place many text components to proper text line later on. The proposed line segmentation algorithm can draw a straight as well as curved boundary. The results of initial segmentation [109] on a merged text lines page is depicted in Fig. 3.19(a). CLS algorithm of the proposed line segmentation algorithm is applied on the same page and resultant initial segmented page is depicted in Fig. 3.19(b).



a) Line segmentation by [110] line segmentation algorithm

b) Initial line segmentation by the proposed line segmentation algorithm

Fig. 3.21 Comparison of line segmentations of [110] and the proposed line segmentation algorithm

The line segmentation [109] segments a text page into zones and subsequently zones are segmented into lines by applying straight split line. So, whenever CLS algorithms changes its path for drawing boundary, actually it avoids the later processing tasks as compared to line segmentation in [8]. For example, two words and in line 9 and 11, respectively, are correctly segmented by CLS algorithm but [109] is incapable of such initial segmentation, therefore, it necessitates to process these tasks later on.

Furthermore, red circled tasks in Fig. 3.20 (a) are still there to be resolved after initial segmentation by [109], yet the proposed algorithm resolves more than half of them encircled as blue shown in Fig. 3.20(b).

A recent line segmentation algorithm [110] also used the zone based method as defined by [8] with additional processing of dilating for better estimation of zone size.

Fig. 3.21(a) represents the initial segmentation process and there is still need to resolve some overlapping issues later on. On the other hand, output of the proposed algorithm has perfectly segmented the page into text lines and there is no need for further processing as depicted in Fig. 3.21(b).

(a) Initial segmentation of first page (b) Final segmentation of first page (c) Initial segmentation of second page (d) Final segmentation of second page (e) Final segmentation of second page by [29]

Fig. 3.22 Urdu Nastaleeq pages taken from [104] and segmented by propose line segmentation algorithm

Table 3.4
Statistics about Components of UPTI Dataset

	Primary components		Secondary components		Total components	Ligatures segmented	Segmentation accuracy
	Touching baseline	Not touching baseline	Touching baseline	Not touching baseline			
UPTI dataset undegraded	**176353**	**12879**	**18801**	**124459**	**332492**	**189232**	**99.80%**

A state of the art contribution [104] for layout of Urdu and Arabic script that is actually an improvement of [103], which has depicted two very complex sample Urdu Nastaleeq pages as shown in Fig. 3.22 (a) and (c).

Table 3.3
Initial Split Row Numbers

Table 3.3(a) Initial split for Fig. 3.22(a)

Lines #	Intensity Value	Split Row Number
1&2	42	93
2&3	83	164
3&4	82	248
4&5	79	325
5&6	45	390
6&7	36	483

Table 3.3(b) Initial split for Fig. 3.22(c)

Lines #	Intensity Value	Split Row Number
1&2	3	61
2&3	19	120
3&4	27	178
4&5	24	224
5&6	14	273
6&7	24	328
7&8	10	383
8&9	17	433

The complexity of the pages may be observed by more interlacing and merging of components of subsequent lines. The average intensity of initial split line positions are 46 and 14 for first and second page, respectively, as presented in Table 3.3. The highest intensity values at split positions pose maximum obstacles in drawing boundary, whereas, a straight line can demarcate a boundary between two text lines in case of minimum

intensity value (i.e. 0). The proposed line segmentation algorithm is applied on these two complex pages that extracts text lines accurately with only one diacritic out of place encircled red as shown in Fig. 3.22(b). Furthermore, some encircled dots/diacritics are misplaced by algorithm of [104] as depicted in Fig.22 (e).

The proposed line segmentation algorithm is also applied on an Arabic language page provided in [104]. The proposed line segmentation algorithm misplaced only one diacritic, while line segmentation algorithm of [104] misplaced 5 diacritics as shown in Figs. 23(b) and 23(c), respectively. Lines are segmented by both algorithms with 100% accuracy.

3.5.3 Results of the proposed ligature segmentation algorithm

UPTI data set [103] is used for checking the segmentation accuracy of the proposed segmentation algorithms. This data set contains images of 10063 sentences. Four degradation techniques that were proposed in [111] are applied, which include jitter, elastic elongation, threshold and sensitivity. Every degradation technique is applied with four different parameter values. Overall, UPTI contains 12 degradation versions of original 10063 sentence images. The sentence images of UPTI data set from un-degraded version is segmented into ligatures as presented in Table 3.4.

There are total of 189584 ligatures as per split of ground truths of UPTI data set. Accuracy of the implemented algorithms is even better than the best ligature segmentation accuracy reported in [109] i.e. 99.02%.

Fig. 3.23 Arabic Nastaleeq page taken from [104] and segmented by propose line segmentation algorithm

UPTI data set contains incorrectly merged ligatures that badly affect the accuracy, because the corresponding ligatures in ground truths are not properly connected. Therefore, count of image ligatures reduces as compared to ligatures segmented from text lines of ground truth, resulting in low accuracy. Examples of these incorrectly connected dots, diacritics and ligatures are depicted in Fig. 3.24.

First  and second  part of Fig. 3.24 represent two ligatures connected incorrectly. Fourth part  consists of all secondary components of third part  and represents six dots connected incorrectly. The overall size of these connected dots becomes more than the maximum threshold size of a secondary component, also it is touching the baseline. Therefore, ligature segmentation algorithm declares it as a primary component.

Table 3.5
Statistics about Components of First Sentence of UPTI Dataset

Primary comp. touching baseline		Secondary comp. touching baseline		Total comp.	Ligatures segmented
Yes	No	Yes	No		
21	6	4	20	51	27

Table 3.6
Comparison of Urdu Language Ligature Segmentation Algorithms

Ligature segmentation algorithms	Primary components			Secondary components			Total comp.	Ligatures	Accuracy
	Total	No confusion	With confusion	Total	No confusion	With confusion			
Javed et al. [108]								Corr./total 3375/3655	92%
Din et al. [110]								6811/7364	99.49%
Lehal [109]	23555			18886	17565	12805	42441	23555	99.02%
This study	189232	176353	12879	143260	18801	124459	332492	189232	99.80%

The proposed algorithm is tested for segmenting un-degraded version of the UPTI data set. It is very interesting that the overall secondary components are less than the primary components. Further, the ratio of primary to secondary components is about 7:5. There are total of 189,584 ligatures as per split of ground truths of UPTI data set. Ground truths don't have any incorrect connectivity, so, in this way UPTI data set contains 189,584 ligatures and the proposed algorithm extracts 189232 ligatures as mentioned in Table 3.4.

Fig. 3.24. Wrongly connected components of ligatures

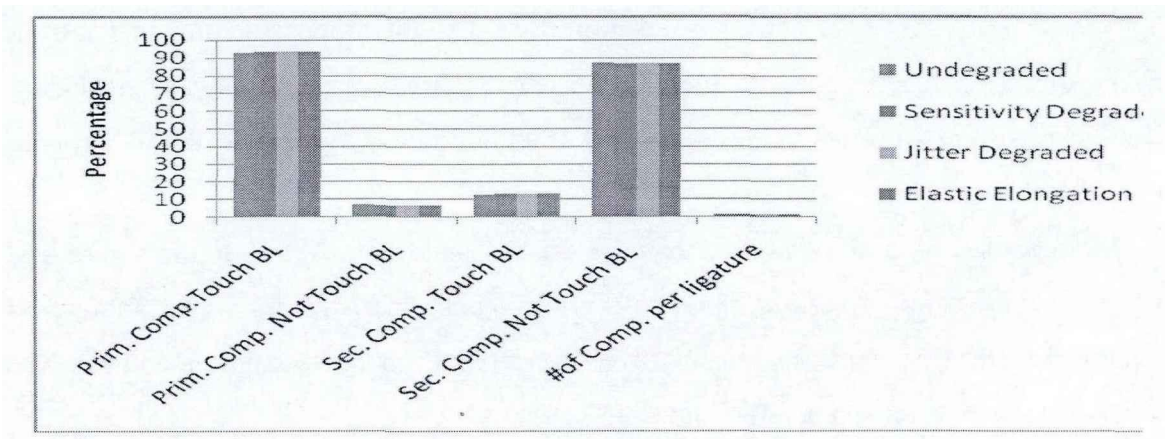


Fig. 3.25. Percentage statistics about components of UPTI data set

Table 3.4 indicates that about 189000 ligatures are successfully extracted from 10063 text lines consisting of approximately more than 332000 connected components from UPTI data set. Furthermore, total of about 143000 secondary components are allocated to approximately 189000 primary ligatures with an accuracy rate of 99.80%.

Fig. 3.25 depicts the percentage statistics of ligature segmentation algorithm for four versions of UPTI data set. The primary components touching the baseline are approximately 93%. Similarly, the secondary components that are not touching the baseline are approximately 87%, while the rest are not touching the baseline. Besides, every ligature consists of approximately 1.7 connected components. Table 3.5 represents observed statistics by applying the proposed ligature segmentation algorithm on first sentence of UPTI data set.

Table 3.6 presents the results of segmentation algorithms and their accuracy along with the sizes of data sets used. The primary components that are touching baseline with size greater than the threshold size of a primary component are considered as primary components without confusion and rest are with confusion. Similarly, conflicting (with confusion) and non-conflicting (no confusion) secondary components are explained in Algorithm 5.

Javed et al. [108] used total 3655 ligatures, having 3375 correctly segmented ligatures with 92% accuracy, as presented in first row of Table 3.6. Israr et al. [110] tested their algorithm on approximately 7364 ligatures with 99.49% accuracy. Lehal [109] tested his algorithm on 23555 ligatures with 99.02% accuracy. Last row of Table 3.6 depicts the evaluation of the proposed algorithm on first version of UPTI data set, as presented in Table 3.4. Also, 10063 text lines consisting of more than 332000 connected components from every subset of data set were extracted. Furthermore, total of about 143000 secondary components were allocated to approximately 189000 primary ligatures with the accuracy rate of 99.80%.

The proposed algorithm is better than the existing state of the art algorithms. The best ligature segmentation algorithm [109, 110] are using six heuristics for segmentation into ligatures. Instead of heuristics, quantitative values (width, height, centroids and baseline touching) are used for segmenting lines into ligatures.

3.6 Conclusion

This chapter mainly introduces two algorithms for line and ligature segmentation of Nastaleeq text images. The proposed line segmentation algorithm places dots and diacritics more accurately as compared to [103, 104]. Prevailing work [109, 110] relied more on zonal information and heuristics for line and ligature segmentation, respectively. This approach relies more on baseline for decision making as opposed to zonal information for line segmentation. Also, proposed algorithm employs baseline, width, height, centroids, and overlapping quantitative information for ligature segmentation algorithms.

The ligature segmentation algorithms is tested on a very large data set with improved accuracy as presented in Table 3.4. On average, for UPTI data set, the proposed algorithm segmented 189000 ligatures from 10063 text lines with 332000 connected components. A total of about 142000 secondary components have been successfully allocated to more than 189000 primary ligatures. The results of the proposed line and ligature segmentation algorithms with accuracy of 99.17% and 99.80% respectively, are thus, better than the precision realized by the prevailing Urdu Nastaleeq segmentation algorithms.

3.7 Summary

The underlying chapter put forward two effective line and ligature segmentation algorithms for Nastaleeq Urdu document images. For this purpose, the projection profile method with a novel Curved Split Line algorithm is explained for segmentation of text lines of Urdu document images in detail with algorithms. Further, a ligature segmentation algorithm is explained that successfully segmented text lines of UPTI dataset with an accuracy of 99.08%. Overall, both proposed segmentation algorithms performed comparable with the state of art segmentation algorithms for Urdu document images. After, successfully applying segmentation to Urdu document images, now it is possible to apply learning algorithms for the recognition of ligatures. Next chapter details about a ligature recognition through deep learning method.

CHAPTER 4

Stacked Denoising Autoencoder based Urdu Nastaleeq Ligature Recognition

Offline Urdu Nastaleeq text recognition has long been a serious problem due to its very cursive nature. In order to get rid of the character segmentation problems, many researchers are shifting focus towards segmentation free ligature based recognition approaches. Majority of the prevalent ligature based recognition systems heavily rely on hand-engineered feature extraction techniques. However, such techniques are more error prone and may often lead to a loss of useful information that might hardly be captured later by any manual features. Most of the prevalent Urdu Nastaleeq test recognition was trained and tested on small sets.

4.1 Introduction

During the last two decades, a lot work has been done on character based [41-43, 45, 46, 61] and ligature based [11, 24, 47, 48, 50, 59, 109] Urdu OCR. Despite of their success, there remain two major limitations: lack of suitable feature extraction techniques and scarcity of available data. Almost all researchers have applied hand-engineered feature extraction techniques which may not fully represent the data. Secondly, the largest reported dataset used comprised of approximately 83000 ligatures [60], which is not large enough for evaluating practical Urdu OCR.

This chapter addresses two main shortcomings that currently Urdu OCR systems exhibit. Firstly, autoencoder is used for feature extraction method which is already proved very successful [70-74] for recognizing many image datasets by extracting features automatically from raw pixel values. And secondly, training is done by using 178573 ligatures from un-degraded version of UPTI dataset [59] and tested on almost same number of ligatures from degraded versions of UPTI with 93% to 96% accuracy.

4.1.1 Prevalent Works

Previous works done on OCR for Printed Urdu Nastaleeq text can be classified into two types: characters based method and ligature based method. In literature, the former is known as segmented oriented while the latter as segmented free techniques also. A ligature may be a word or a sub-word which is composed of one or more connected characters. A ligature may usually compose of 1 to 8 characters. The Nastaleeq text recognition, as compared to Naskh, is more complex due to its overlapping, compactness, context sensitiveness, cursive nature, diagonality, and many other[14] properties shown in Fig. 4.1.

Initially, considerable efforts were devoted to the recognition of Urdu basic character set. MLP Network [41], SVM [45], NN [43], HMM [61], PCA [46], Back Propagation (BP) based NN [40], and pattern matching classifier [42] have been used for recognition of Urdu characters. All of these OCR systems have used man designed features like invariant moments [45], structural features (width, height and checksum) [43], Discrete Cosine Transforms (DCT) [61], topological features [40] and chain code [42]. The largest reported dataset is 36800 characters [45] with the accuracy of 93.59%.

Difficulties and complexities [14] of Urdu Nastaleeq text segmentation into characters cause the paradigm shift from character to ligature based Urdu text recognition systems. Most of the prevalent Urdu OCR techniques work on ligature based recognition[11, 15, 24, 47, 48, 59, 60, 109].

Ligature based OCR systems produce superior results as compared to the counterpart

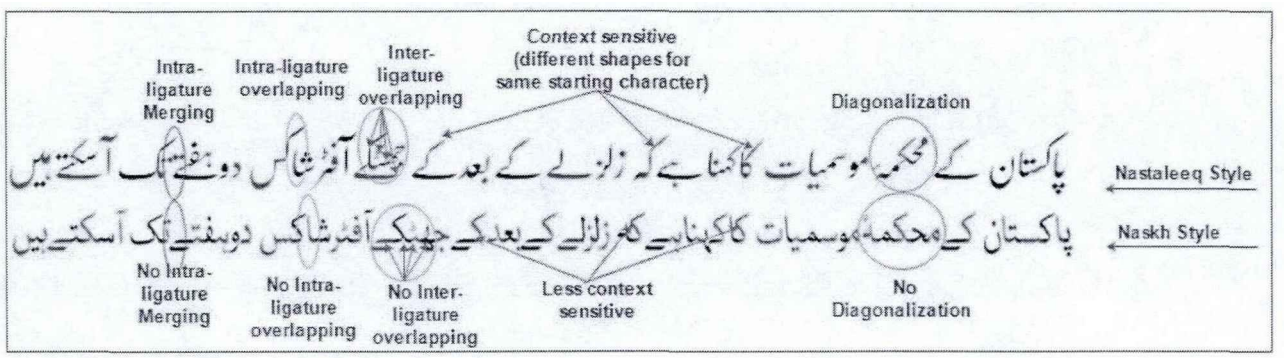


Fig. 4.1 Comparison of Nastaleeq and Naskh writing styles

character based recognition schemes.

Various models and features are explored in literature on ligature based Urdu text recognition. Many choices of learning models have been used for this purpose, such as

HMM [11, 15, 24, 47, 60], KNN [59,116], SVM [109], and BLSTM structure [48, 50] with a layer of CTC at output. From the previous work, it is evident that the DCT is the most dominant features extraction technique used [11, 15, 24, 47, 60, 109] for Urdu OCRs. Some other features include shape context [59], raw pixel values [48], and sliding window [50], and Hu invariant moments [116], which are all man-designed. In Urdu OCR systems, no one has so far used any of the deep learning methods e.g. autoencoders for automatic feature extraction [71].

4.1.2 Motivation

Scarcity of data for training is one of the major drawbacks of all Urdu OCR systems to-date. The largest data set for Urdu OCR system consists of 83000 ligatures [60]. Also, one can easily detect that all available Urdu OCR systems are heavily relying on hand-engineered feature except [48, 50]. Such features are very hard to design, expensive, domain dependent and do not accurately mimic the properties of a dataset. Manual labeling like [48, 50] of an enormous data set is almost impossible.

Autoencoders have been used to extract feature representation directly from the raw pixel values of an image [71, 72, 114]. With noisy input, Denoising Autoencoders (DAs) have been used to extract robust hidden representation of data independent of domain [70, 74, 115]. It can represent the intrinsic features of data which are very difficult to uncover by human-designed features.

Stacked Denoising Autoencoders (SDAs) have been successfully applied for character recognition [73] and recognition of Bangla Language [112, 113]. Therefore, with the success of automatic feature extraction by autoencoders in many image recognition fields, it is a reasonable selection to let SDAs learn features itself from raw data of Urdu Nastaleeq text.

4.1.3 Contribution

In this chapter, a ligature based Urdu OCR application is presented by using the basic SDA [73]. No one in Urdu OCR research, thus far has reported the use of DA or SDA for feature learning. Here, different SDA networks with softmax layer on top are trained and

tested on a very large dataset, known as UPTI dataset [59]. Detail about UPTI is presented in Section 4.1.

In the referenced work, the largest data set for Urdu OCR system consists of 83000 ligatures (224 pages *371 ligatures per page) [60]. Similarly, over 10,000 ligatures have been used in [59] from UPTI data set. Although [42] reports 98% accuracy, it uses very small data set i.e. noise free 9262 ligatures of 2190 classes.

URL-SDA is trained on 178573 ligatures from 3732 classes of un-degraded version of UPTI and almost same number of ligatures from degraded versions for validation and testing. So, here training and testing set is more than twice in size than the largest dataset [60] used for Urdu OCR, so far. Validation and testing the trained SDA gives 4.14% error which is also lesser than the reported error in the referenced work.

4.2 Learning model

This section presents the introduction of basic autoencoder, denoising autoencoder and the architecture of proposed SDA model for ligature recognitions of Urdu language. Then the Algorithm 1 illustrates the training procedure of proposed SDA model.

4.2.1 Autoencoder

The most well-known use of deep neural networks is feature learning [71-74]. It offers a way to learn useful feature vector for the posed data [75]. It may learn a compact, meaningful representation of the posed data, typically aiming for reducing dimensions [72].

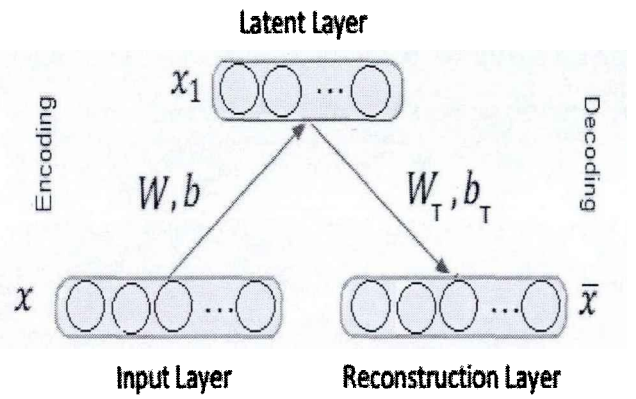


Fig. 4.2 Basic autoencoder structures

Autoencoder is basically a feed-forward and non-recurrent neural network. There may be one or more hidden layers in between input and output layer as shown in Fig. 4.2. An autoencoder is a MLP except that it has equal number of inputs and output nodes. Moreover, an autoencoder predicts the input value x at the output. It does a feed-forward pass and predicts the value of $\bar{x}$, measures the difference between x and $\bar{x}$ and back propagates by performing weight updates.

If the hidden layers are less than the input/output layers, then the final hidden layer represents the compressed representation of the input [76]. As mentioned earlier, autoencoder is a subtype of MLP, therefore it can use all the activation functions used in MLP.

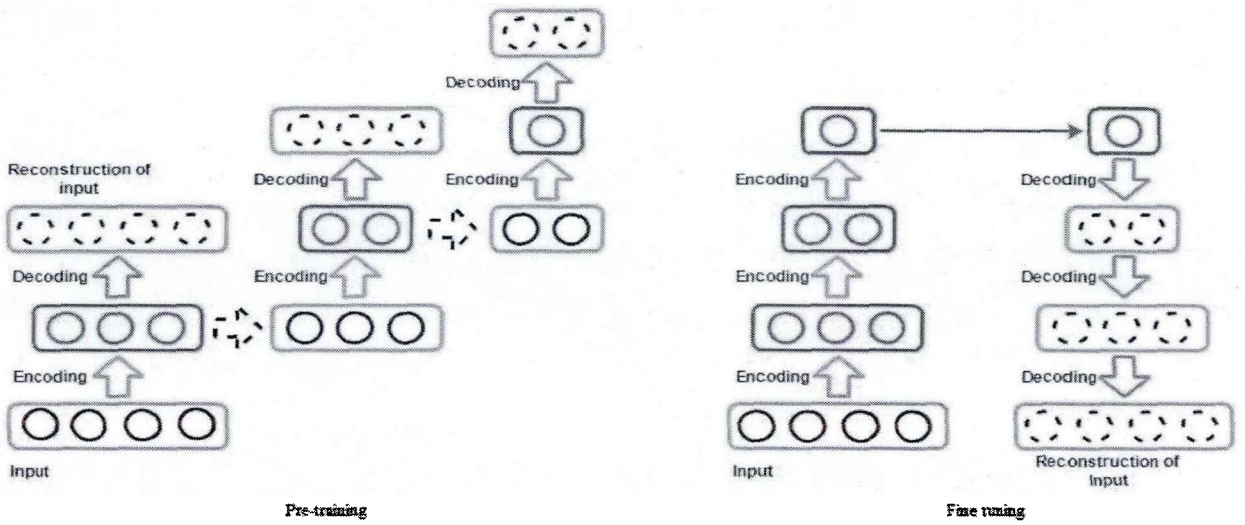


Fig. 4.3 Training process of Denoising Autoencoder

Denoising Autoencoder is a type of autoencoder, which tries to learn the input x at the output as $\hat{x}$ by working on partially corrupted version of input. In this way, the learnt $\hat{x}$ may be more stable, robust and gives better higher level representation [70].

Different methods are available for corruption of input data, like MN, a suitable proportion of image pixels are masked to 0; Salt-and-Pepper Noise (SP), where randomly selected pixels of an image are set to maximum and minimum values of the image uniformly [73]. The MN corruption method is used in this work for experiments.

4.2.2 Urdu Ligature Recognition Stacked Denoising Autoencoder (ULR-SDA)

In this work, denoising autoencoders is used for learning Urdu ligature features. The architecture of the Stacked Denoising Autoencoder is illustrated in Fig. 4.3. SDA network training consists of two steps: pre-training and fine-tuning. Former is performed in unsupervised way and latter in supervised manner. All layers of SDA are first trained layer wise, getting the input from latent representation of the previous network except the first hidden layer, which gets its input from outside. Pre-training is very helpful in initializing the network nodes by good representation instead of initializing them randomly [71]. A MLP is made of all pre-trained layers and a back propagation algorithm is used for fine-tuning the network.

The process of the proposed Urdu Ligature Recognition Stacked Denoising Autoencoder (ULR-SDA) is illustrated in Algorithm 1. ULR-SDA also follows the two stage general setup of SDA as already explained. A deep stack of denoising autoencoders network is first pre-trained on the images of ligatures in an unsupervised manner, layer by layer as illustrated by first for-loop in Algorithm1. Afterwards, the trained layers are connected together to form a MLP. At the top of this MLP network, a logistic regression layer is added that uses softmax function for classification. This resultant network, ULR-SDA is then fine-tuned to anticipate the target ligatures as depicted by pseudo code lines 16-27 of Algorithm1.

Algorithm 1. Pseudocode for Training ULR-SDA

Input :

L = Layers of the Stacked Denoising Autoencoder

D = List of the numbers of Nodes for each layer

X = Ligature training set, one example per row

Y = Target output ligature class for each corresponding row of X

B = Batch size for training

C = List of corruption levels for each layer L

lr_pre_trian = Learning rate for pre training

lr_fine = Learning rate for fine tuning

Output: NetParam = $\{\text{NetParam}^i\}_{i=1}^{L}$, weight and biases of the trained SDA network

Pseudocode:

```
1.  for i =1 to L do
2.      Randomly initialize NetParam^i = D_{i-1}, D_i
3.      b = sizeof( X^i ) / B //number of training batches
4.      epoch = 0
5.      repeat
6.              for j = 1 to b
7.              epoch = epoch+1
8.          X^j = get_batch(X^i, b)
9.          y_j = activation_function (NetParam^j, get_corrupted(X^j))
10.         x̄_j = get_reconstructed_input( y_j )
11.         l_j = X^j log x̄_j + (1 - X^j) log x̄_j   //loss by cross entropy
12.         update NetParam^j with stochastic gradient of cost l_j with its parameters
13.       endfor
14.     until mean ( l ) > 0.1 % or epochs < number_of_pretrain_epoch
15. //now fine tune the model
16. epoch = 0
17. b = sizeof(X)/ B
18. repeat
19.     for j = 1 to b
20.     epoch = epoch+1
21.             X^j = get_batch(X, b)
```

22. y_j = activation_function ($NetParam^j$, $get\_corrupted(X^j)$)
23. $\bar{x}_j$ = get_reconstructed_input(y_j)
24. $l_j = X^j \log \bar{x}_j + (1 - X^j) \log \bar{x}_j$ //loss by cross entropy
25. update, backpropagate $NetParam^j$ by stochastic gradient of cost l_j with its parameters
26. endfor
27. until validation error > 0.05 % or epochs < number_of_finetune_epoch
28. return(NetParam)

The flowchart of proposed ULR-SDA is represented by Fig. 4.4. It consists of three basic steps, namely, preprocessing, training, and testing. Preprocessing performs segmentation of image and ground truth sentences into ligatures. Training is carried out according to Algorithm 1. Testing performs classification with accuracy.

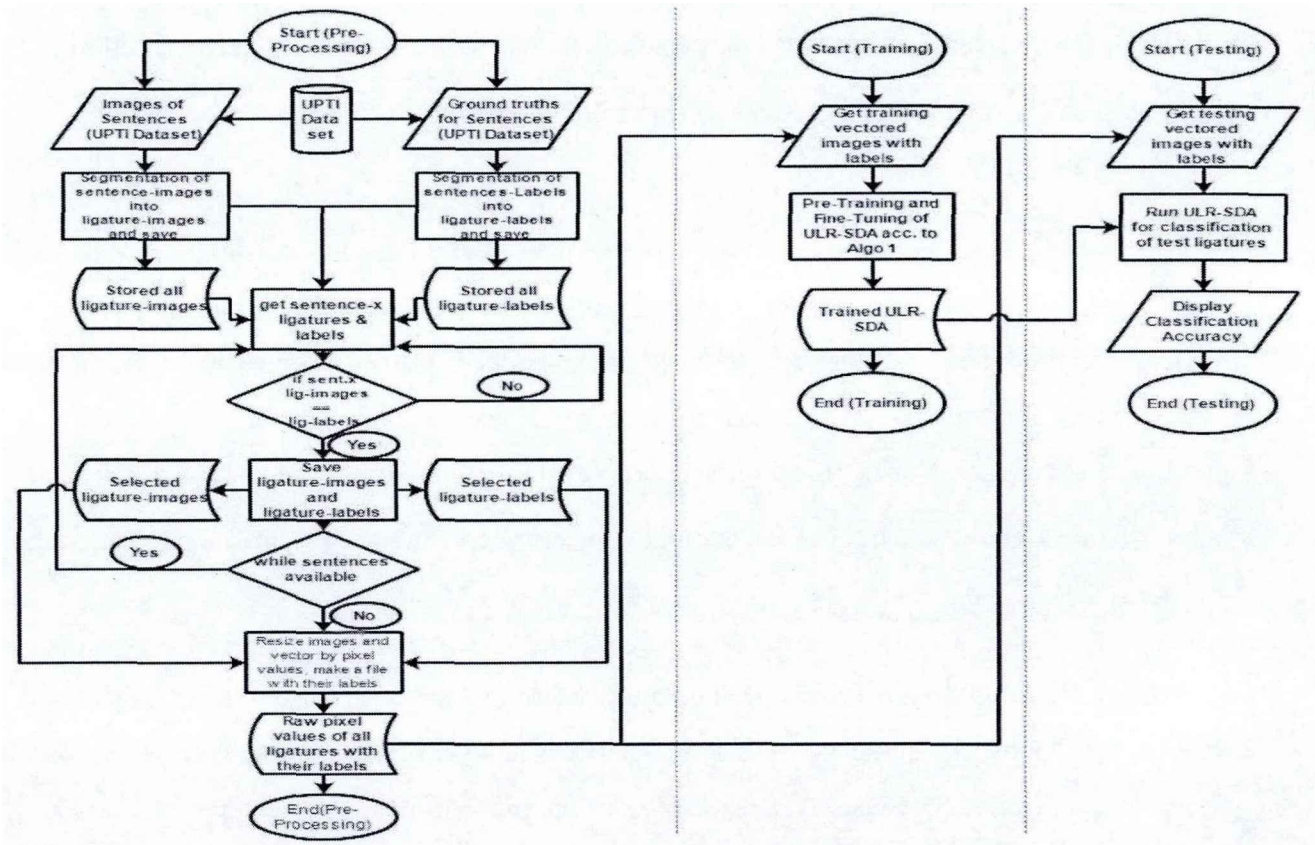


Fig. 4.4 Preprocessing, training and testing process of Urdu Ligature Recognition Stacked Denoising Autoencoder (ULR-SDA)

As proposed by Vincent et al. [71], for pre-training the stack of denoising autoencoder, the deep networks can be learnt using denoising autoencoders. The underlying model is trained with the intent to learn the hidden representation for reconstructing the input as normally done by standard auto encoder network. The only difference is that the denoising autoencoder is fed with the noisy input with the objective to learn more generalized hidden representation.

For this purpose, the conventional approach of Masking Noise (MN) [73] is employed in this model to which the random pixels values are set to zero. For getting the hidden representation y, a non-linear logistic function $s(x)=\frac{1}{1+e^{-x}}$ on weight matrix W with the input x and the bias vector b is calculated as follows:

$$y = s(Wx + b) \tag{4.1}$$

Here, the tied weight W and the bias vector b are used for encoding. The reconstruction $\bar{x}$ is computed by using transposed weight matrix W^T, transposed bias vector b^T and non-linear function s as follows:

$$\bar{x} = s\left(W^T y + b^T\right) \tag{4.2}$$

Thus, the derived autoencoder is trained on the noisy version of the input ligature x for its reconstruction. Therefore, the next autoencoder is trained in the same fashion, but the input for training of the next autoencoder is the hidden representation of the previous autoencoder. During this process, the reconstruction error between non-corrupted input ligature x and reconstructed $\bar{x}$ is computed by cross-entropy as presented in [71]:

$$L_H(x,\bar{x}) = -\sum_{i=1}^{d} x_i \log \bar{x}_i + (1-x_i)\log \bar{x}_i \tag{4.3}$$

Once all autoencoders are pre-trained, they are connected layer wise to form a feed-forward MLP network. At the top of this MLP network, a layer of logistic regression added for the classification. The last layer uses softmax activation function for estimating class probabilities. This resulting complete MLP network of denoising autoencoders is fine-tuned through backpropagation algorithm.

4.3 Experiments

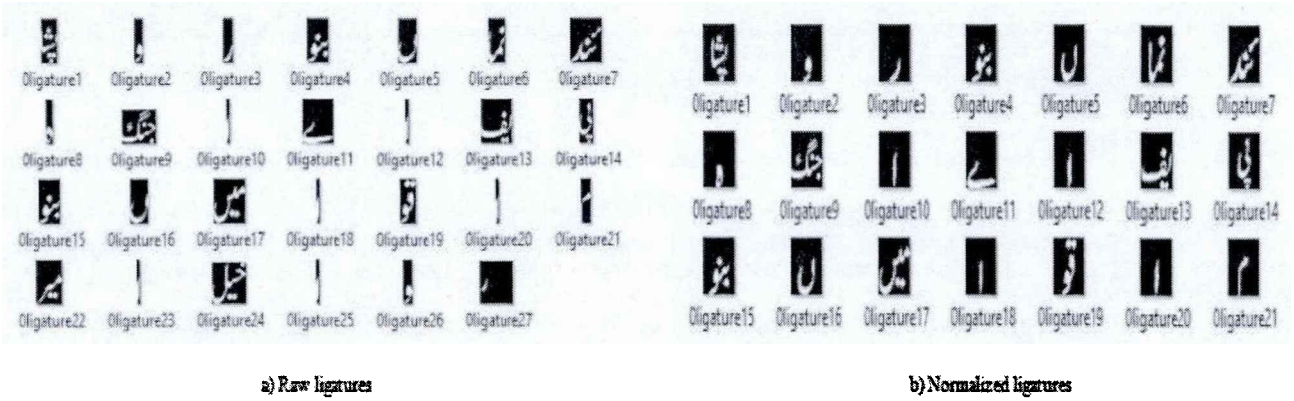


Fig. 4.6 UPTI segmented ligatures

This section discusses the experiments for evaluating ULR-SDA architecture on printed offline Urdu script.

Two types of experiments based on different input dimensions were performed on UPTI dataset. ULR-SDA was trained on 178573 ligatures and tested on almost same number of ligatures with 3732 classes for 80*80 and 15*15 input dimensions.

4.3.1 Dataset and Feature Extraction

UPTI[7] dataset [59] is used for experimental purpose. This dataset contains 10063 sentence images. A sample sentence of UPTI is shown in Fig. 4.5.

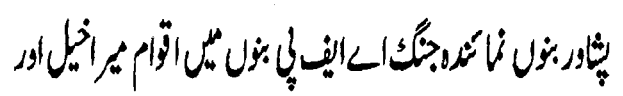


Fig. 4.5 UPTI sample sentence

Four degradation techniques [111] have been applied, namely: jitter, elastic elongation, threshold and sensitivity. Every degradation technique has been applied with four different parameters value. So, UPTI contains 12 degradation versions of original 10063 sentence images. The sentence images of UPTI dataset from un-degraded as well as jitter, elongation and sensitivity degraded versions were segmented into ligatures as shown in Fig. 4.6. The number of ligatures of un-degraded, jitter degraded, sensitivity degraded,

[7] UPTI (Urdu Printed Text Images) dataset is provided by faisal.shafait@uwa.edu.au and adnan@cs.uni-kl.de

and elastic elongation degraded versions are 189262, 189265, 189260, and 189262 respectively. Segmented ligatures are then resized according to requirement as shown in Fig. 4.6(a, b).

The ground truths are then tokenized into ligatures, having 3732 number of tokenized ligature classes after segmentation. Only ligatures from sentences are considered valid where a sentence contains equal number of images ligatures and label ligatures. This is because of the irregularities induced during printing as shown in Fig. 4.7. Two ligatures are incorrectly connected by the extra length of the strokes, for example ضاگیلا and آگنا. Fourth part consists of all secondary components of third part. Fourth part represents eight dots, out of that six are connected incorrectly. Size of these connected dots become more than the maximum threshold size of a secondary component and assumes as primary component by segmentation algorithm. Because of such incorrect connections, number of the image ligatures becomes less or more in numbers as compared to ligatures segmented from text lines of ground truth.

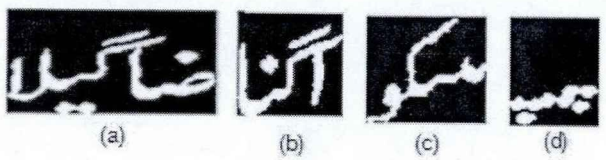


Fig. 4.7 Wrongly connected components of ligatures

Only 3732 unique Urdu ligatures have been used in the UPTI dataset after dropping out the sentences where any sort of irregularity exists, as illustrated in preprocessing stage. Resized and normalized images of ligatures are then vectored into their pixel values and stored with their labels in tuple form for feeding as input to ULR-SDA. SVM classifier was also trained and tested with this format.

4.3.2 Experiment Setup

For performing the experiments, ULR-SDA is trained on 178573 ligatures of un-degraded version and 60,000 each from jitter degraded and sensitivity degraded versions of UPTI dataset for validation and testing respectively. There are 3732 classes of Urdu

ligatures8, which are used in UPTI dataset. Two different versions of UPTI dataset, i.e.; ligature images of 80*80 and 15*15 dimensions, have been constructed for experiments. For comparison study, a SVM classifier has also been trained and tested as explained earlier for ULR-SDA. In all cases, unsupervised pre-training and supervised fine-tuning (with simple stochastic gradient descent) procedures were applied, with early stopping based on validation set performance. In Fig. 4.8 and 9, the "h" stands for 100 and 'k' for 1000. For example in Fig. 4.8, 25h-16h-9h and 7k-5k-4k stand for SDA with hidden layers [2500, 1600, 900] and [7000, 5000, 4000] respectively.

4.3.3 Training ULR-SDA

Different ULR-SDA networks are trained and tested for ligature recognition as shown in Table 4.1 and Fig. 4.8. Best ULR-SDA network has three hidden layers with [7000, 5000, 4000] units for 80*80 input dimensions and [2500, 1600, 900] units for 15*15 input dimensions. All hidden layers were pre-trained as denoising autoencoders by stochastic gradient descent, using the cross-entropy cost method with the learning rate of 0.001.

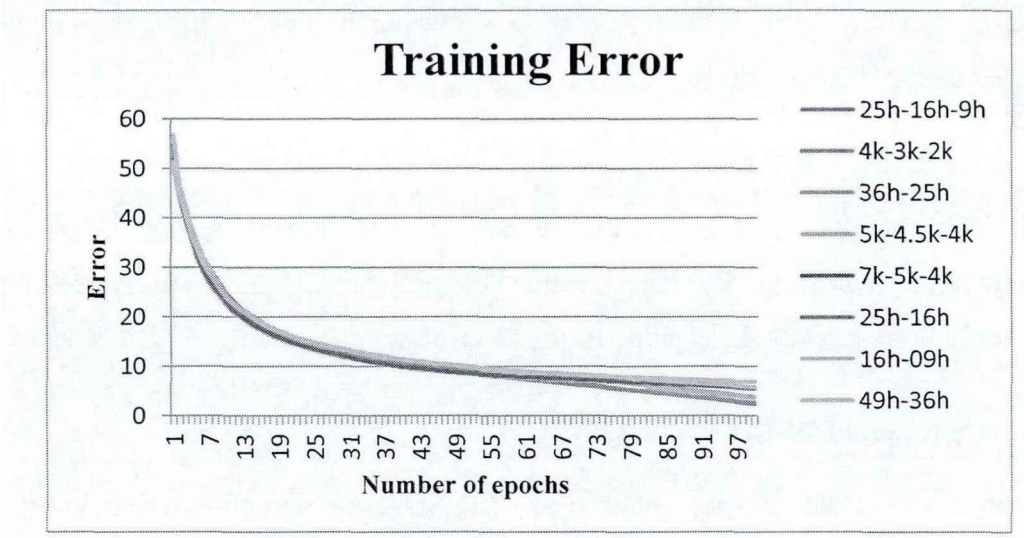


Fig. 4.8 Number of epochs and training error of ULR-SDA

[8] A repository of 18,000 Urdu ligatures:
http://www.cle.org.pk/software/ling_resources/UrduLigatures.htm

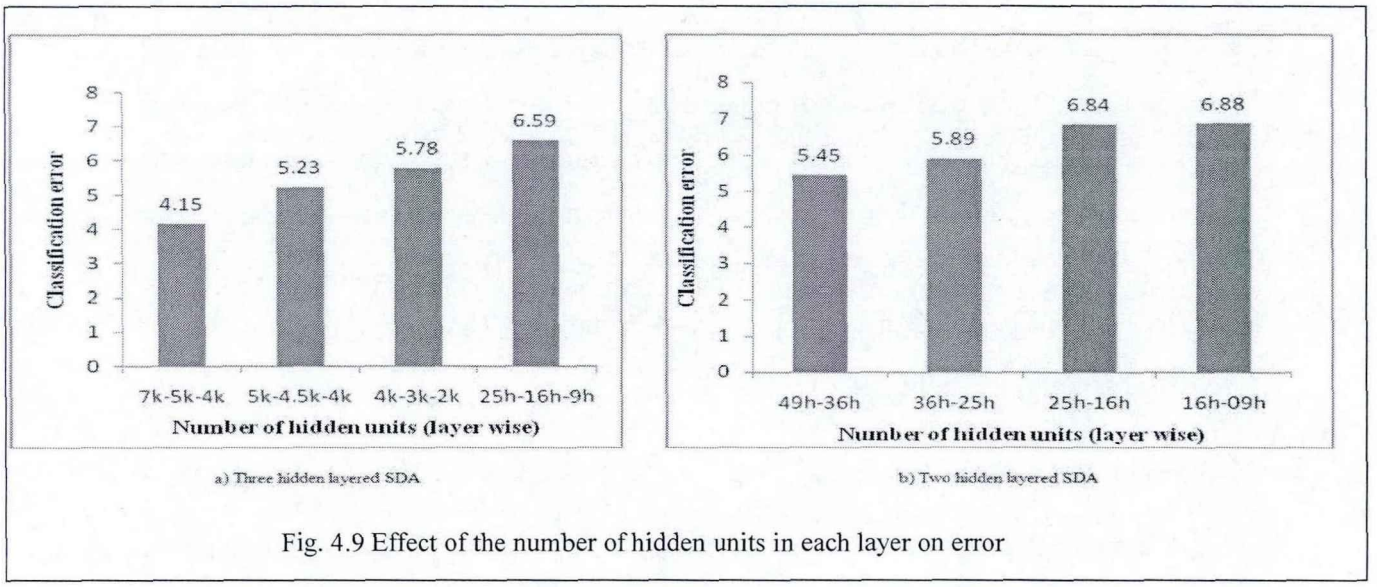


Fig. 4.9 Effect of the number of hidden units in each layer on error

For all experiments during pre-training, 10 epochs were executed for ligatures of 80*80 as well as for 15*15 dimensions. Fine tuning of the entire MLP network was done by stochastic gradient descent using cross entropy loss function while adopting the learning rate of 0.1. The fine tuning executed until number of epochs were less than100 or the validation error did not fall below 0.1 % as shown in Fig. 4.8. All experiments were conducted using Theano library 6 on GPU.

4.4 Results

In this section, the results of ULR-SDA and SVM are compared. After the comparison, the analysis of different structures of ULR-SDA is presented. At the end, the effects of input and middle layer dimensions on error are discussed.

4.4.1 Comparison of ULR-SDA and SVM

In order to evaluate the performance of ULR-SDA, the system is trained on the un-degraded ligatures of UPTI dataset. The trained network is then tested on elastic elongation, jitter and sensitivity degraded versions of the same dataset.

Table 4.1 Comparison of URL-SDA and SVM Based on Same Training and Test Data

Table 4.1(a) Results for 80*80 pictures

Algorithm	Training and Testing Setup	Accuracy
ULR-SDA	Layers:[7000,5000,4000], learning rate=0.001	96%
SVM	SVM Multi class, gamma = 0.01	95%

Table 4.1(b) Results for 15*15 pictures

Algorithm	Training and Testing Setup	Accuracy
ULR-SDA	Layers:[7000,5000,4000], learning rate=0.001	95.86%
SVM	SVM Multi class, gamma = 0.01	85%

For comparison purpose, a multiclass SVM with Radial basis function(RBF) kernel is trained and tested in the similar fashion as ULR-SDA. Both classifiers namely ULR-SDA and SVM are trained on 80*80 and 15*15 input dimensions for 3732 output classes. Three layered ULR-SD achieved the ligature recognition accuracy of 96% where the SVM got 95% accuracy for 80*80 input dimensions as shown in Table 4.1(a). Similarly, three layered ULR-SDA recognized ligature with the accuracy of 95.86% and SVM got 85% accuracy for 15*15 input dimensions Table 4.1(b). The ULR-SDA shows superior recognition result as compare to state of the art SVM algorithm as shown in Table 4.1. Furthermore, both classifiers achieved better accuracy on 80*80 input dimensions as compared to 15*15, which is obvious because former contains more pixel information of every ligature as input.

4.4.2 Structure of ULR-SDA

The structure of the proposed ULR-SDA is evaluated on two parameters: firstly by increasing the number of neurons of hidden layers and secondly by increasing the number of hidden layers as are shown in Figures 9 and 10 respectively.

For this evaluation, 15*15 input dimensions have been used for all networks. Fig. 4.9(a) shows the performance deteriorates with the decrease in the number of hidden units per layer. Three layered network [7000, 5000, 4000] performs better than the network with hidden layers [5000, 4500, 400], [4000, 3000, 2000], and [2500, 1600, 900]. Similarly, Fig. 4.9(b) shows two layered network [4900, 3600] and [3600, 2500] performs better than [2500, 1600] and [1600, 900].

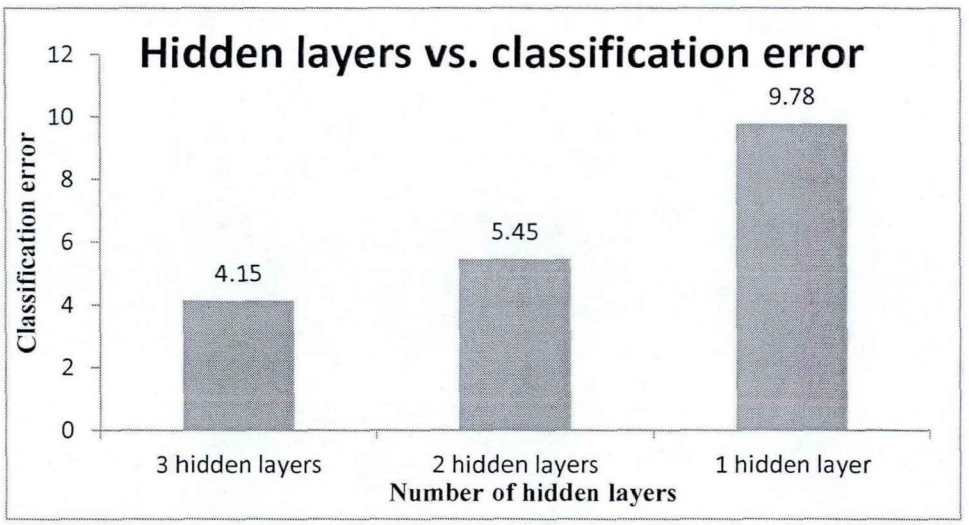


Fig. 4.10 Effect of the number of hidden layers on error

Fig. 4.10 shows the increased performance of ULR-SDA as the increase in the number of hidden layers from 1 to 3, for three different networks. Apart from Fig. 4.10, Fig. 4.9 also shows that the performance of the 3 layered networks is better than that of two layered networks.

4.4.3 Dimensions

The performance of ULR-SDA varies by input data dimensions and dimension of middle layer as shown in Table 4.2 and Fig. 4.11. ULR-SDA has been trained on examples

of 80*80 and 15*15 dimensions. ULR-SDA outputs better results when trained and tested on examples of 80*80 input dimensions as shown in Table 4.2.

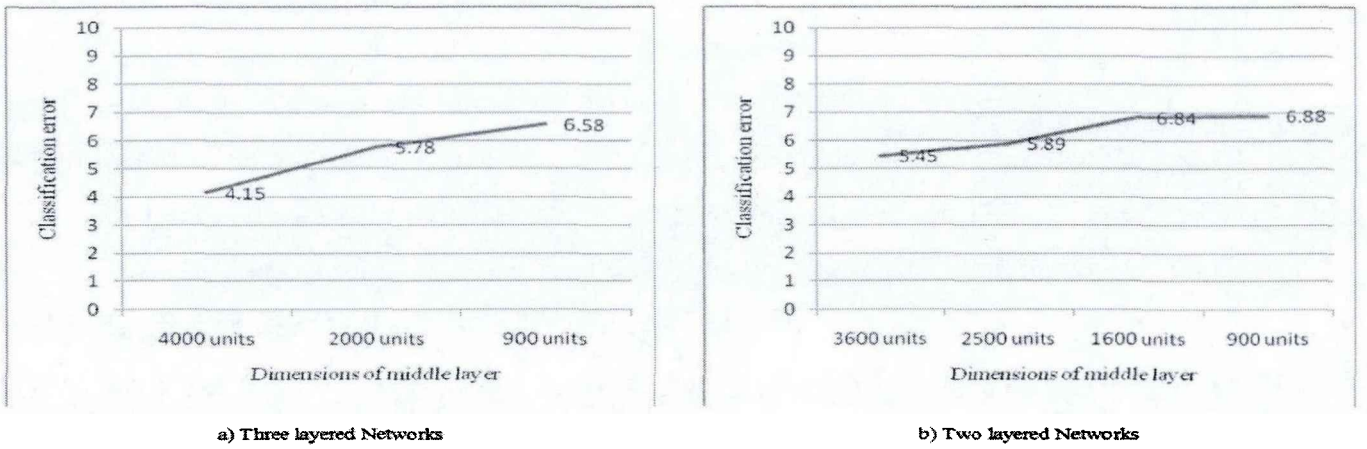


Fig. 4.11 Dimension of middle layer and error

Table 4.2: Effect of Number of Input Data Dimensions on Accuracy (3732 classes, 180k Training Examples)

Table 4.2(a) Results for Hidden Layers[7000,5000,4000]

Configurations	Classification Accuracy
Input dimension:80*80	96%
input dimension:15*15	95.86%

Table 4.2(b) Results for Hidden Layers[5000,4500,4000]

Configurations	Classification Accuracy
input dimension:80*80	96%
input dimension:15*15	94.77%

The recognition accuracies of ULR-SDA upon training and testing the examples of 80*80 and 15*15 dimensions with hidden layers [7000, 5000, 4000] are 96% and 95.86% respectively as shown in Table2(a).

Similarly, the recognition accuracies for 80*80 and 15*15 input data dimensions with hidden layers [5000,4500,4000] are 96% and 94.77% respectively as shown in Table2(b). The accuracy will be increased with the increase in dimensions of input data.

In Fig. 4.11(b), the general trend is showing an increase in error for two layered network as well. More informative middle layer representation results in better accuracy in URL-SDA can, therefore, be confirmed.

4.5 Conclusion

This work is inspired by the success of recent SDA deep networks for characters and digits recognition. At the same time, the recent work on SDA for recognition of Bangla Language [112, 113] motivated us towards the implementation of deep SDA for Urdu Nastaleeq recognition. This works mainly introduces the use of deep neural networks (DNN) in the field of Urdu OCR. Also, it demonstrates that the process of recognition may become very straightforward and easy, if raw pixel values are used instead of calculating different features. In this way, better representation of all aspects of input data may be efficiently achieved, which may enhance the performance of existing Urdu OCR systems.

In this chapter, stacked denoising autoencoders and softmax at output layer have been used for automatic feature extraction directly from raw pixel values and classification respectively. Different stacked denoising autoencoders have been trained on 178573 ligatures with 3732 classes from un-degraded UPTI data set. The trained networks are then validated and tested on degraded versions of UPTI data set. To compare the performance of ULR-SDA, multi class SVM classifiers have been trained and tested on the same train and test sets. Test results show that the accuracies of ULR-SDA networks are 93% to 96%, while the accuracies of SVM classifiers are in the range of 80% to 95%. The results of ULR-SDA based ligature recognition achieved are therefore better than the precision realized by the prevailing Urdu OCR systems for such a large dataset of ligatures.

4.6 Summary

This chapter introduces a ligature based Urdu text recognition system named as ULR-SDA. This system extracts features automatically using deep learning stacked denoising autoencoder and softmax is used for classification. The ligatures are extracted from UPTI datasets by employing ligatures segmentation algorithm discussed in chapter 3. ULR-SDA outperformed the state of art SVM algorithm. During the recent years, another deep learning methodology, LSTM has got considerable attention from pattern recognition and

machine learning community. LSTM architecture is used for learning with context. It has already been successfully applied for Urdu sentence recognition. The subsequent chapter introduces an effective ligature based Bidirectional LSTM methodology for UPTI sentence recognition.

CHAPTER 5

Ligature based Urdu Nastaleeq Sentence Recognition using Gated Bidirectional Long Short Term Memory

BLSTM architecture - a special case of RNN - is successfully applied for recognition of Urdu Nastaleeq sentence images based on character information. In such cases, manual labeling of characters in sentences for a large dataset is an intensive job, because identical characters observe different shapes at different positions inside ligatures and words. On the other hand, labeling any dataset with ligatures is a relatively easier and more accurate phenomenon. In the current chapter, a novel GBLSTM model for recognition of printed Urdu Nastaleeq text based on ligature information. The proposed model incorporates raw pixel values as features instead of human crafted features, because of the latter being more error prone. The model is trained on un-degraded and tested on unseen artificially degraded versions of UPTI dataset. The recognition accuracy of the proposed GBLSTM model is 96.71% that is higher than the prevalent Urdu OCR systems.

5.1 Introduction

Urdu is the national language of Pakistan, and is widely spoken and understood in the Indian subcontinent [13, 117]. According to the frequency compiled list of languages [10], Urdu falls among top ten most influential languages of the world. Urdu is written mainly in Nastaleeq writing style. It has a rich collection of published material, and amount of Urdu documents is increasing steadily. Nastaleeq OCR, therefore, has great demand and receives more and more attentions.

Urdu OCR systems are primarily categorized as segmentation free (holistic approaches) and segmentation based (analytical approaches). Segmentation free approaches try to learn and recognize shapes of the entire ligatures [15, 16, 31, 59, 119, 120]. Segmentation based techniques mainly rely on segmenting text into characters for text recognition [11, 29, 47- 53, 118].

In Urdu script, there is no definite inter word spacing, therefore, it is hard to extract and subsequently learn words directly from text [14]. Urdu words are composed of one or more ligatures. Every ligature is separated from other ligatures with a space. So, recognition of

words could be possible by learning ligatures followed by post processing that incorporates dictionary validation [50]. A ligature is composed of one or more characters that can easily be extracted from text as compared to characters. Segmentation free OCR methodologies [15, 16, 31, 61, 119] mainly rely on recognition of ligature.

Segmentation based classifiers mainly rely on segmenting text into characters. Segmentation can be explicit [11, 47, 118] or implicit [29, 48-53]. However, it is hard to segment Nastaleeq text into characters accurately due to very cursive nature of Urdu Nastaleeq text [14]. Moreover, characters recognition becomes more difficult because of its context sensitivity, compactness, overlapping, diagonality and several other characteristics [6-8] as shown in Fig. 5.1.

In implicit segmentation based classifiers, text images with corresponding labels are fed to the system, which in turns try to identify characters based upon segmentation cue points [29]. Such recognition approaches have the advantage of learning with context or sequence property instead of instance learning [11, 15, 16, 31, 47,118-120]. These classifiers are used to learn sequence of language units and are applied in Urdu language for sentence recognition. The LSTM based recognition methodologies [29, 48-53] rely on implicit segmentation. The BLSTM captures bidirectional context that includes information about previously given input and upcoming input. The BLSTM was first introduced by Alex Graves [121] and later successfully applied for Urdu Nastaleeq recognition [29, 48-53]. In such systems, labeling sentences with correct form of characters is a difficult task. The segmentation of ground truth sentences into ligatures for labeling and formation of sentences from recognized ligatures are simple processes as compared to processes defined in prevalent works [29, 48-53].

5.1.1 Motivation

Ligature based OCRs [15, 16, 31, 59, 119, 120] are more accurate as compared to character based, but can't extract context information that is important for making semantically correct sentences and paragraphs. Character based LSTM models [29, 48-53] perform well with better accuracy level due to availability of contextual information. However, labeling input sentences with correct form of characters is a difficult process.

This problem can be addressed by applying a novel idea of ligatures based LSTM model, as labeling at ligature level is more convenient.

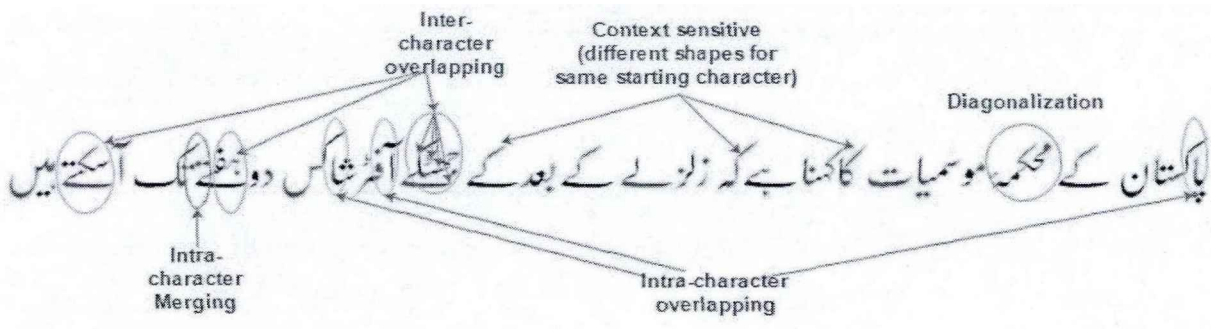


Fig. 5.1 Difficulties of Nastaleeq writing styles

Urdu character set consists of joiner and non-joiner characters [51]. Non-joiner characters are "ا, د, ڈ, ذ, ر, ڑ, ز, ژ, و, ں, and ے", that always appear in isolation or at final position of a ligature. Joiners characters of Naskh can be written in four different shapes namely initial, middle, final, and isolated [118], as presented in Fig. 5.2. In case of Urdu Nastaleeq writing style, the number of possible shapes of some characters may even approach to 60 [62, 63].

The process of sentences labeling with characters described by [29, 48-53] is difficult for a large dataset like UPTI, as visualized in Fig. 5.3.

Sentence labelling with characters by [48] is depicted in Fig. 5.3(a) with each character having a unique label and a suffix. Characters are divided into 8 categories of 99 basic and 191 all shaped characters. Generally, non-joiner characters are suffixed by 3, except noon-gunna (ں) that is always suffixed by 0. Joiner characters are labelled by suffixing 0, 1 and 2 based upon their position of occurrence in a ligature. Joiners at initial and second positions in ligature are suffixed by 0 and 1, respectively. Joiners from third to second last position in ligature are all suffixed by 2.

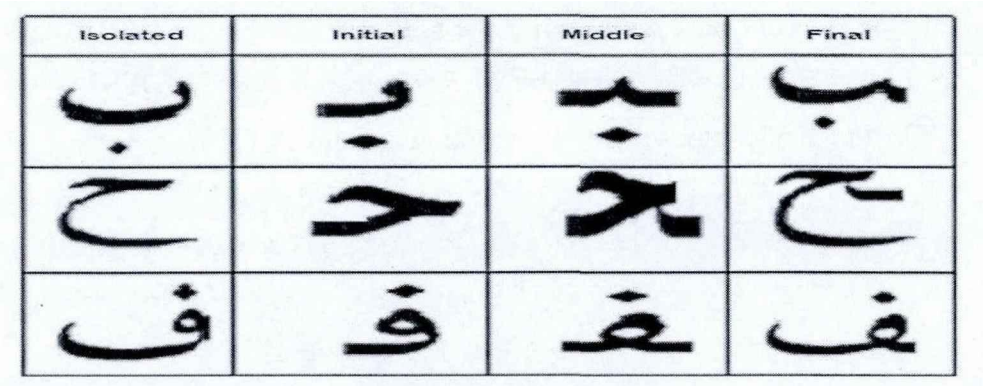


Fig. 5.2 Four different shapes (Isolated, initial, middle and final) of 3 (joiner) characters

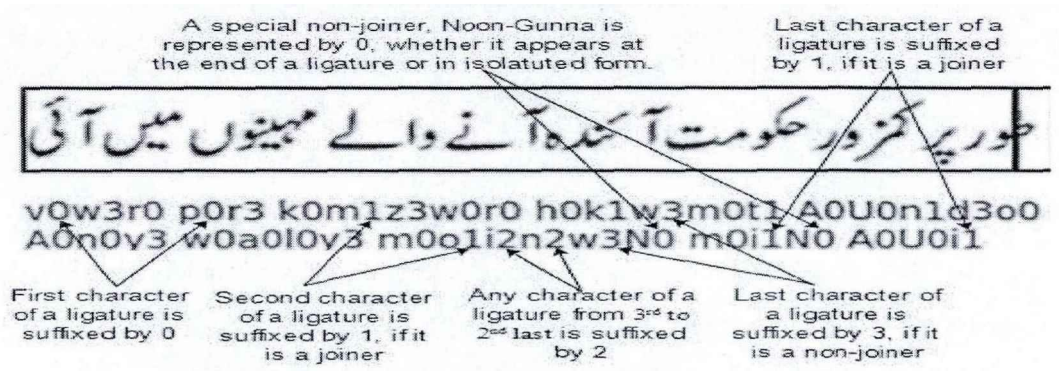
The labeling process of [29, 49-53] is shown in Fig. 5.3(b) and 3(c). Words in a sentence are separated by space, while character labels of words are separated by hyphens. Total characters used for labeling process are 43. The sample labeled sentence contains two characters 'ے' and 'ی' that are represented by same label 'yea' as depicted in Fig. 5.3(b). Fig. 5.3(c) shows that middle form of 'ی' is represented by a label 'yea' and rest all occurrences of 'ے' and 'ی' are represented by a same label 'yeh'. Representation of different shaped/class characters with such labels will ultimately create confusion during recognition time. Therefore, it is an intensive task to label sentences with correct form of character, ranging from 43 to 191 classes.

Features are extracted from sentence, and then fed to system for recognition as described in [29, 49, 50, 52]. However, human crafted features are more error prone and are proved [71, 72,125] to be less optimal for representing every property of the objects to be recognized.

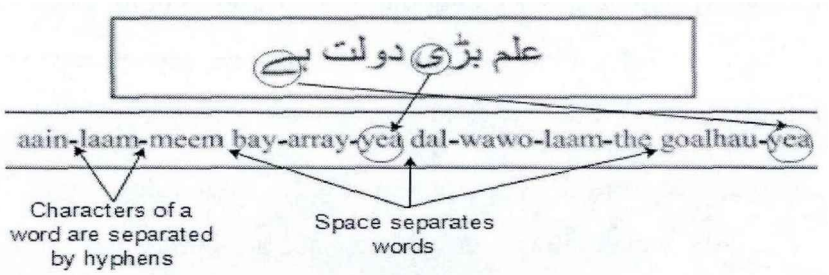
Furthermore, accuracy of OCR systems [29, 48-53] highly depends on selection of optimal window size. Sliding window is an approach to extract features of fixed height and width from an image of a sentence that are fed to the recognition system. Although smaller window size provides better results, but it takes more training time as compared to larger window size [49]. A sliding window of 30×1 is used in [48, 51]. Similarly, a sliding window of size 4x48 and 4 different sliding windows(2x48, 4x48, 6x48, 8x48) are employed in [49] and [50], respectively. Additionally, BLSTM [48, 51] and MDLSTM

[29, 49, 50, 52, 53] systems are using a CTC layer for input to output alignment. Maximum number of character classes recognized in the referenced work [48] are 191 and 91 with accuracies of 86.4% and 94.8%, respectively. The latest proposed MDLSTM based OCR systems [29, 50, 52, 53] achieved accuracies ranging from 94.77%

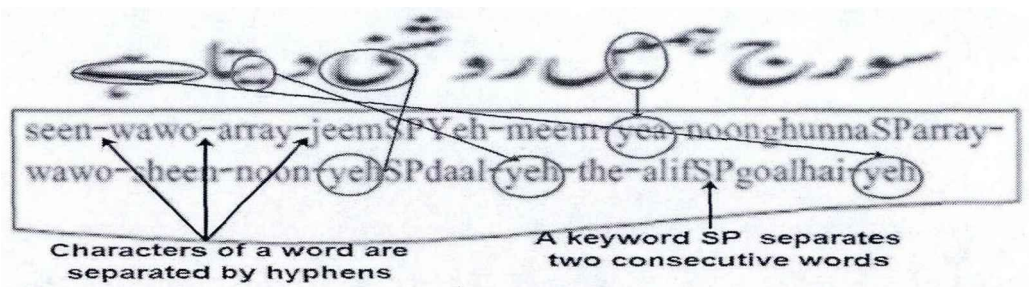
to 98±0.25% for 43 classes of characters.



(a)



(b)



(c)

Fig. 5.3 Labeling sentences of UPTI dataset by existing systems, Labeling sentences of UPTI dataset (a) by [48], (b) by [29, 49-52], and (c) by [53]

The proposed ligature based GBLSTM structure, with pre segmented and aligned input doesn't need CTC layer and performs comparable to existing BLSTM and MDLSTM structures for Urdu text recognition.

5.1.2 Contribution

Recognizing contextual information is a challenging task for developmental Urdu OCRs. Earlier contributions on Urdu OCRs are mostly based on recognition of segmented characters or ligatures [11, 15, 16, 31, 47, 59, 119, 120]. This work proposes a novel ligature based GBLSTM for Urdu Nastaleeq text recognition based on contextual information.

Labeling sentence images with correct form of Urdu characters is a difficult task as certain Urdu Nastaleeq characters may adopt shapes that may approach to 60 [62, 63]. On the other hand, labeling sentences with ligatures is comparatively easier as their shapes remain same anywhere inside the text. For labeling purpose, UPTI dataset is segmented into ligatures with an accuracy of 99.08% by applying ligature segmentation algorithm [137].

In this work, Gated BLSTM led to state-of-the art performance for sentence recognition due to its multiplicative gate units' interaction between outputs of forward and backward layers. The role of multiplicative gate unit is to collect common information from output of forward and backward layers.

Furthermore, the proposed GBLSTM model neither needs to select any window size nor CTC layer. Moreover, the proposed GBLSTM model recognizes 3604 ligature classes with an accuracy of 96.71% on UPTI dataset.

5.2 Overview of Recurrent Neural Network and Related Work

HMM paved way for use of context (sequence) information through transition probabilities [126]. Subsequently, sequence of features were extracted from images and transcribed by HMM. It was simply replaced by RNN [127] that could be trained directly on raw images. ANN based RNN were used for the recognition of time dependent data/patterns' context information [128]. RNN recognizes features of images with their sequential binding aspects [121]. Vanishing gradient problem of RNN is its limitation to

retain context of information for long time. This problem was solved by LSTM network [97], which has the ability to access and retain long range temporal context.

The LSTM remarkably outperformed in comparison to HMM [49, 129-132] for character and speech recognition. Bi-directional Recurrent Neural Network (BRNN) [133] is a more robust contextual information retainer network that has the ability to retain information in forward and backward directions, as depicted in Fig. 5.4.

The LSTM cells are incorporated as building blocks in BRNN network and it becomes BLSTM network that superseded RNN, LSTM, and HMM-RNN on phoneme recognition [124].

The BLSTM networks does not work in case of no alignment between input sequence and output labels. For this alignment purpose, a CTC layer [138, 122] was added as an output layer of BLSTM network. This layer trains network to calculate conditional probability distribution for all possible classes of data.

All BLSTM based Urdu Nastaleeq text recognition systems [29, 48-53] were evaluated on UPTI dataset. BLSTM was proposed for printed Urdu Nastaleeq text recognition on the basis of raw pixel values by [48]. Sentences were labeled in two ways: labeling by only base form of characters (99 classes) and labeling with all forms of characters (191 classes), thus, achieving accuracies of 86.4% and 94.8%, respectively. BLSTM [51] employs raw pixels for recognition of text lines. The reported accuracy for UPTI dataset was 88.94%. Two MDLSTM based Urdu sentence recognition systems [49, 50] evaluated on UPTI dataset were based on statistical features, with reporting accuracies of 94.97% and 96.40%, respectively. Similarly, a MDLSTM based Urdu sentence recognition systems [52] using zoning features reported accuracy of 93.38%. Another MDLSTM system [29] attained a recognition accuracy of 98%. Naz et al. [53] extracted features using Convolutional Neural Networks (CNN), which are then fed to MDLSTM and recognized sentence of UPTI with accuracy of 98.12% for 43 character classes.

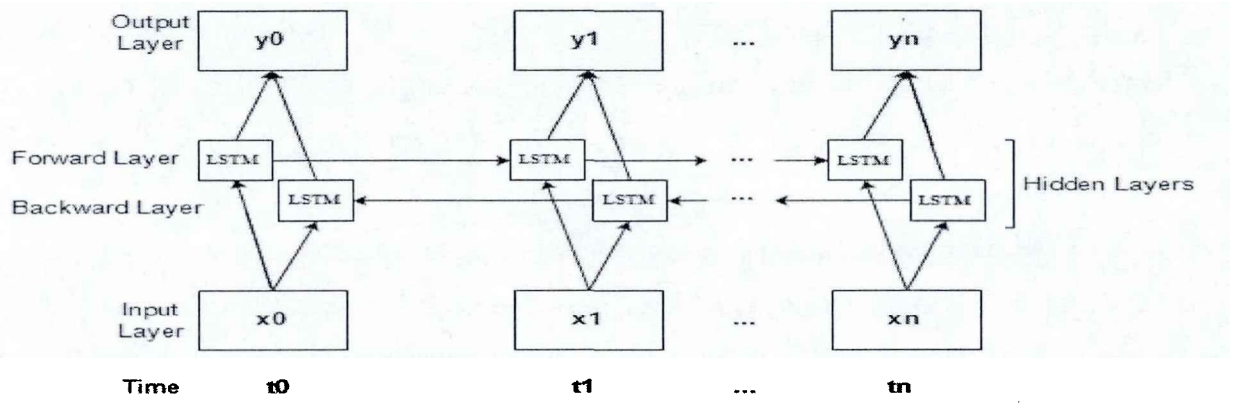


Fig. 5.4 Bi-Directional Recurrent Neural Network (BRNN)

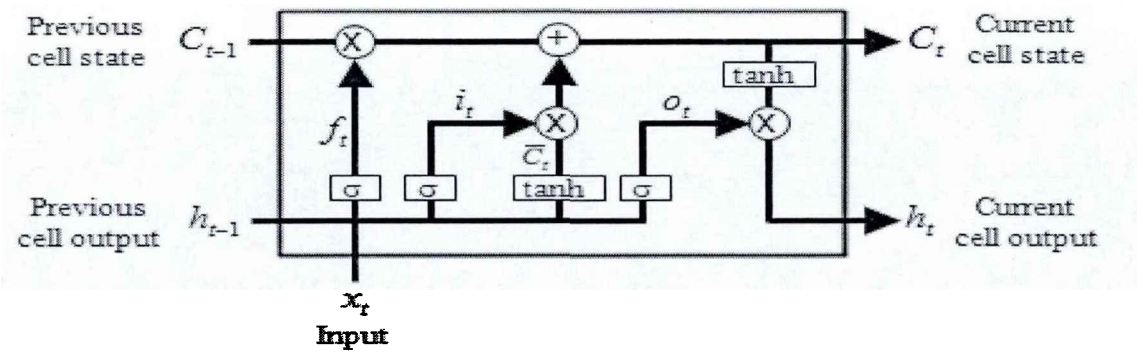


Fig. 5.5 Long Short Term Memory cell: Input and output gates amplify input and output. Current input and previous state of cell is fed to three gates and to memory cell as well. Forget gate layer decides what is needed to retain or to forget. Input gate layer decides to update information for the new cell state. Output gate layer decides which information should be presented at output. Cell state values are used to preserve the context of previous and current input and then output is computed from resulting cell state according to decision of output gate vector.

In the referenced work, LSTM networks have never been evaluated for recognition of Urdu sentences based on ligatures labeling. Furthermore, the accuracy of BLSTM based recognition systems [48, 51] and MDBLSTM [29, 49, 50, 52, 53] depends upon accurate labeling and optimal window size. The maximum and minimum number of character classes recognized in [48] are 191 and 91 with accuracy of 86.4% and 94.8%, respectively. Also, the latest proposed systems [29, 50, 52, 53] are using Multi-Dimensional BLSTM with maximum of 98.12% accuracy for recognition of 43 classes of characters.

5.3 Proposed Model

The proposed ligature based Gated BLSTM model uses BLSTM architecture, where element-wise product of the results of forward and backward layered LSTMs are taken as output. This model is comparatively simpler, as it neither needs CTC layer nor a sliding window. The proposed model is described in details as follows.

A LSTM cell has the ability to access and retain long range temporal context [49, 129-132]. It consists of input, forget, and output gates as depicted in Fig. 5.5. It also preserves its own cell state and an output. These states are updated according to Equations (1)-(7).

$$i_t = \sigma(W_{xi}.x_t + W_{hi}.h_{t-1} + b_i) \tag{5.1}$$

$$f_t = \sigma(W_{xf}.x_t + W_{hf}.h_{t-1} + b_f) \tag{5.2}$$

$$\tilde{C}_t = tanh(W_{xc}.x_t + W_{hc}.h_{t-1} + b_c) \tag{5.3}$$

$$C_t = (f_t \otimes C_{t-1} + i_t \otimes \tilde{C}_t) \tag{5.4}$$

$$O_t = \sigma(W_{xo}.x_t + W_{ho}.h_{t-1} + b_o) \tag{5.5}$$

$$h_t = O_t \otimes tanh(C_t) \tag{5.6}$$

$$p_{t+1} = softmax(h_t) \tag{5.7}$$

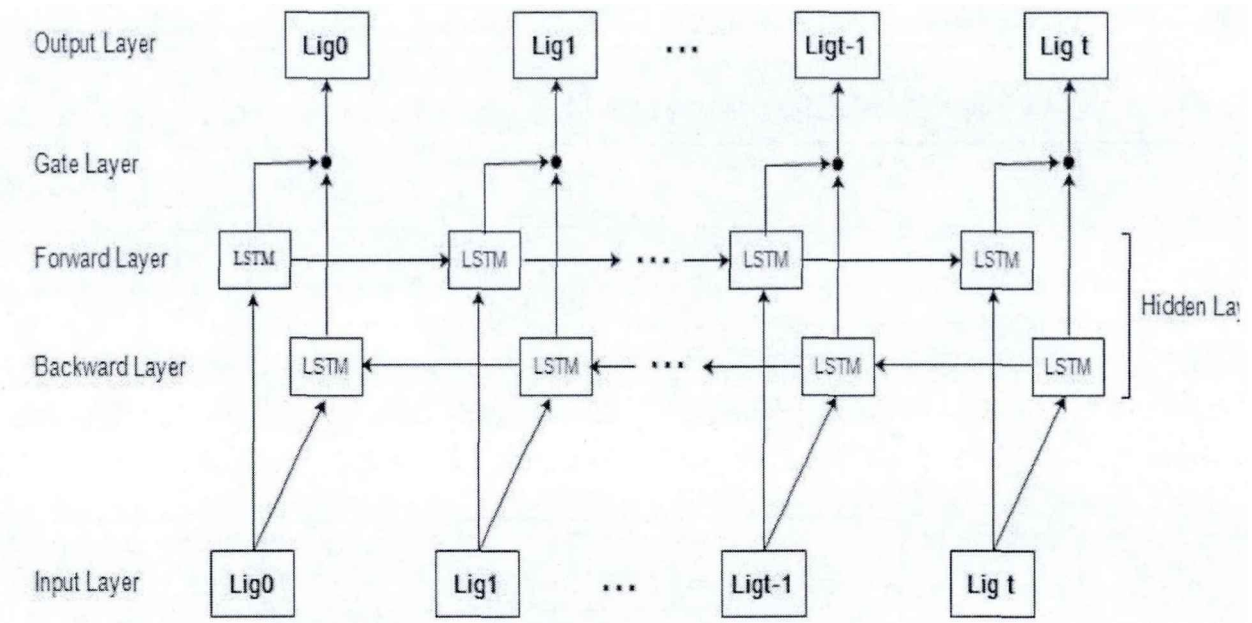


Fig. 5.6 Block diagram of the proposed GBLSTM Model.

Vectors for input gate, forget gate, output gate, cell state and output are represented by i_t, f_t, o_t, C_t, and h_t, respectively. Further, σ is an activation function that can either be ReLU or a sigmoid function. Element-wise product is represented by $\otimes$ that acts as a gate. Distribution of the next ligature is denoted by p_{t+1} that is predicted on the basis of previous ligature context. W_{hi} and W_{ho} are hidden-input gate matrix and input-output gate matrix, respectively. Rest of the weight matrices are cell to gate vectors (e.g. W_{ci}). For interpretation convenience, output of LSTM is presented as follows:

$$h_{t+1} = LSTM(x_t, h_t, C_t) \tag{5.8}$$

Comparing with one directional LSTM, BLSTM considers both previous and following contexts for prediction purpose. Therefore, the output of forward layer cell hF_t at time t is presented as follows:

$$hF_t = LSTM(x_t, hF_{t-1}, CF_t) \tag{5.9}$$

Similarly, output of backward layer cell hB_t at time t becomes:

$$hB_t = LSTM(x_t, hB_{t+1}, CB_t) \tag{5.10}$$

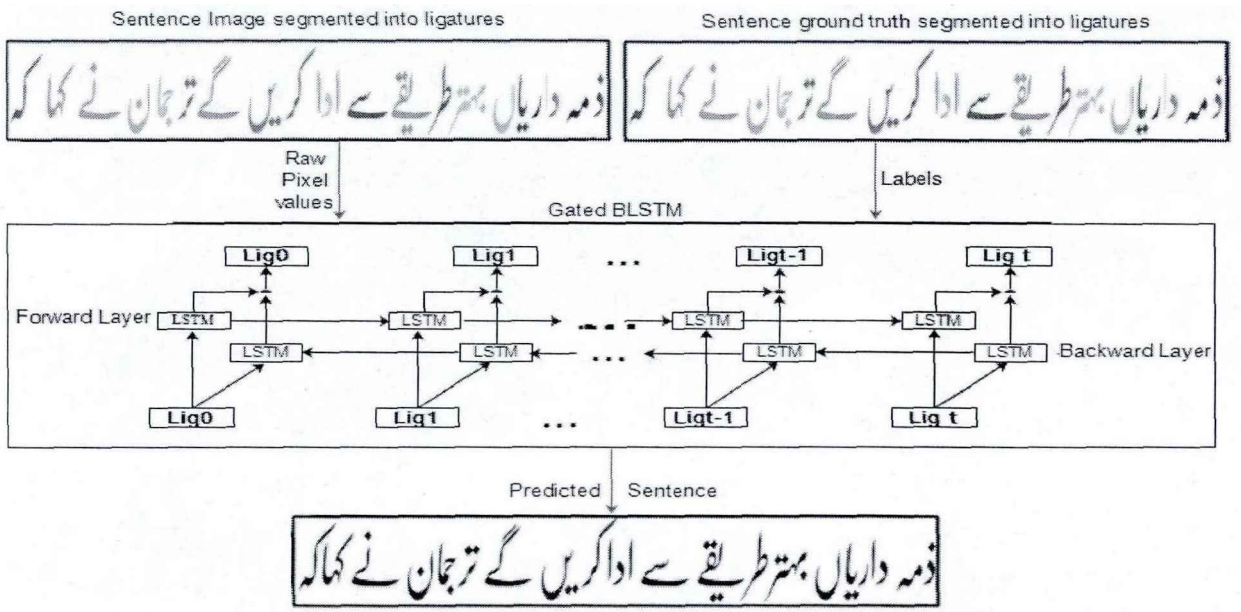


Fig. 5.7 Working of GBLSTM. BLSTM network is input with raw pixel values of ligatures of

The prevalent contributions [16, 18] about BLSTM combine the output of LSTM cell from both directions by adding or concatenating, which preserve all information from both direction for final prediction. In this work, we remark that common information from both directions is required to make better prediction.

The designed gate unit is inspired by the gate unit in LSTM. In LSTM, information of memory cell (C_t) is obtained by combining input signals ($\tilde{C}_t$) and previous cell (C_{t-1}) signals. Each signal is filtered by their corresponding gate units. The gate units produce vectors that are activated by the sigmoid function, which will make each neuron in this vector close to 0 or 1. The element-wise multiplication will let the gate unit filter part of the neurons from $\tilde{C}_t$ and C_{t-1}.

Imitate the gate unit in LSTM, we design a gate unit to filter signals from both forward (hF_t) and backward (hB_t) context. To achieve this goal, we activate both hF_t and hB_t with ReLU, so that the activated neurons are positive numbers and deactivated neurons are 0. Thus, by applying the element-wise production, the activated hF_t and hB_t filter out signals

from each other and only retain the mutual activated neurons that is the joint representation/context from both hF_t and hB_t.

Therefore, the designed gate layer combines information from both directions, which is interpreted by Eq (11) and depicted in Fig. 5.6. We use element wise multiplication to obtain common information. W_d is the output matrix, while $W_{hBg}$and W_{hFg} are referring to backward-gate and forward-gate matrices, respectively.

$$p_{t+1} = softmax(W_d.(ReLU(W_{hFg}.hF_t) \otimes ReLU(W_{hBg}.hB_{t-1}))) \quad (5.11)$$

Table 5.1

UPTI Dataset Splits for Training, Validation and Test Sets

	Training Set	Validation Set	Testing Set	Total
Sentences of UPTI	6800	1600	1600	10000
Ligatures	127180	29924	29935	187039

The presented model also uses ReLU as non-linearity activation function for both forward and backward hidden layers, so that non-zero neurons could be activated and zero-neurons could be deactivated. In this way, common information extracted from both layers adds to the learning activity. Also, ReLU has the advantage over sigmoid and tanh activation function, as it doesn't face gradient vanishing problem [134-136]. By applying element-wise production, common activated neurons are maintained and we consider that this gated vector preserves common information, which is passed to a softmax function for final prediction.

For training, normalized input ligature images of a text line together with ligatures labels are fed to system. Fig. 5.7 depicts that normalized ligature images along with their labels are input to GBLSTM network, which performs forward and backward propagation steps and ultimately predicts the sentence.

5.4 Dataset Description and Preprocessing

Dataset description: UPTI 9 dataset [59] is used for evaluation of the proposed GBLSTM model. UPTI dataset consists of 10063 sentence images and first sample sentence is presented in Fig. 5.8.

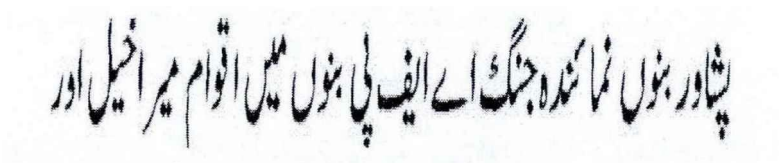


Fig. 5.8 UPTI sample sentence

Fig. 5.9 Segmented ligatures of UPTI first sentence

The dataset contains four degradation versions of original dataset. Degradation is done based on the techniques [111] namely jitter, elastic elongation, threshold and sensitivity. Every degradation method is applied with four different parameter values resulting in 12 different versions of original dataset.

Preprocessing: During preprocessing, images and ground truths of the sentences of UPTI dataset are segmented for feeding to GBLSTM network.

Ligature segmentation algorithm: Sentences are segmented into ligatures by following algorithm of [137]. Numbers of ligatures segmented from first UPTI versions of un-degraded, jitter degraded, sensitivity degraded and elastic elongation degraded are

[9] UPTI (Urdu Printed Text Images) dataset is provided by faisal.shafait@uwa.edu.au and adnan@cs.uni-kl.de

189185, 189251, 189246 and 189244, respectively. All extracted ligatures are then resized to 15x15, 28x28, 60x60, 80x80 and 90x90 dimensions by keeping the aspect ratio of ligature images, as presented in Fig. 5.9.

For finding out the total number of exact ligatures classes, ground truths text lines included in UPTI dataset are also segmented into ligatures. A total of 189584 ligatures as per split of ground truths of UPTI dataset with 3604 classes of ligatures are determined. Total number of selected sentences are 10000 containing 3604 ligature classes. Detail of training, testing and validation split is presented in Table 5.1.

Table 5.2
Network Parameters

Parameters	Value(s)
Learning rate	0.001
Ligature Encoding Size	256
Max Epochs	50
Evaluation Batch Size	100
Hidden Size	256
Train Batch Size	100
Decay Rate	0.999
Drop Prob. for Decoder	0.5
Drop Prob. for Decoder	0.5

5.5 Network Parameters

Performance of the network is optimized by selection of appropriate parameters as depicted in Table 5.2.

Experiments are performed on five types of input ligature dimensions: 15x15, 28x28, 60x60, 80x80 and 90x90. Therefore, sizes of input layers decided by dimensions of input data are: 225, 784, 3600, 6400, and 8100. Rest of GBLSTM network works same for all dimensions of data. Validation is done after every 100 epochs. Several experiments are carried out by varying numbers of hidden layer LSTM blocks from 8, 16, 32, 64, 128, 256

and 512 with constant learning rate of 0.001. It is observed that network performs better as we increase size of hidden layer, but at the same time the training time shoots up as hidden size is also increased from 256 to 512. Thus, final proposed network is trained with forward and backward hidden LSTM layers each of size 256 blocks.

5.6 Experiments and Results

This section discusses experiments for evaluation of proposed GBLSTM model on printed offline Urdu script dataset. Same UPTI dataset is also used for studies done in [29, 48-53] while employing BLSTM and MDLSTM models. Accuracy in the prevalent character based LSTM systems was measured on the basis of number of correctly predicted characters in the test set. Since, we are using ligature based LSTM method, so here system's accuracy depends upon the number correctly predicted ligatures. Number of possible Urdu character classes [29, 48-53] ranges from 44 to 192. On the other hand, number of ligature classes in UPTI dataset is 3604. Therefore, the presented system is needed to learn more classes with less number of training instances for each class as compared to earlier contributions. By using same dataset, we can fairly compare results and performance of the proposed GBLSTM with reported results of prevalent systems. Performance of the network is measured in terms of accuracy that is calculated on the basis of number of correctly predicted ligatures as follows:

$$Accuracy = 100 * (1 - (W_{PL}/T_{SL}))\ ,$$

Where,

$$W_{PL} = \text{Wrongly predicted ligatures}$$

$$T_{SL} = \text{Total ligatures in test set} \qquad (5.12)$$

Five types of experiments based on different input dimensions are performed on UPTI dataset. The proposed system is trained on 6800 sentences with 3604 classes of ligatures and tested on 1600 number of sentences for all 5 input image dimensions. The proposed system is trained on un-degraded version of dataset. For validation purpose, randomly selected 1600 sentences from jitter degraded version are added to training set. Testing set is constructed from unseen 1600 sentences of sensitivity degraded versions of UPTI dataset.

The accuracy of the proposed GBLSTM is almost 96.7% for input dimensions of 60x60, 80x80 and 90x90, as shown in Fig. 5.10. The best results are achieved from 80x80 dimensions.

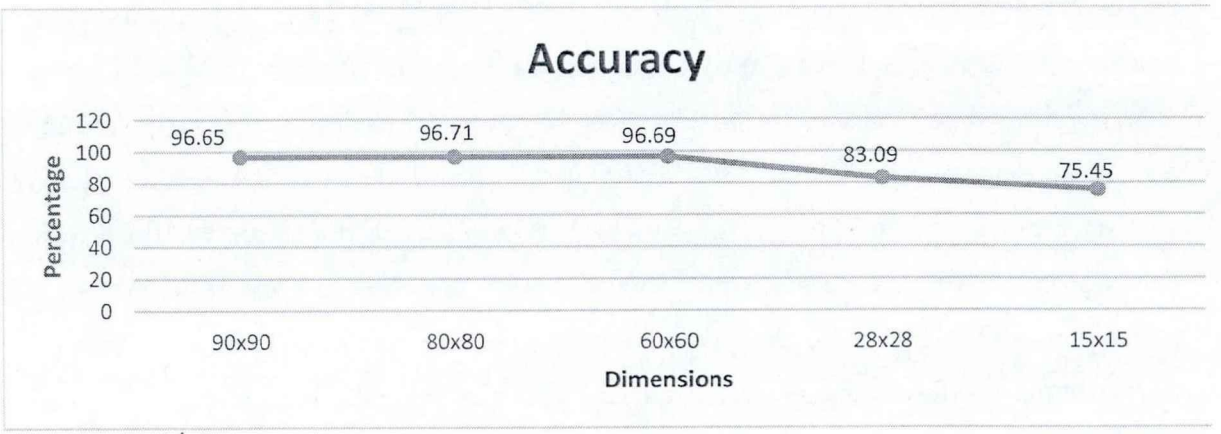


Fig. 5.10 Proposed GBLSTM accuracy comparison on the basis of input dimensions

In subsequent section, the best achieved results from 80x80 input dimensions are compared with the results of prevalent BLSTM and MDLSTM Urdu OCR systems [29, 48-53]. Further, the results of GBLSTM for the input images comprising of five input dimensions are also compared.

5.6.1 Comparison of GBLSTM with Prevalent LSTM based Urdu Recognition Systems

A precise comparison of vital parameters of implemented system with prevalent LSTM based Urdu Nastaleeq systems is presented in Table 5.3. This makes the comparison fair enough on the basis of accuracy as all LSTM based Urdu recognition systems have used the same UPTI dataset except [51]. A super set of UPTI dataset is used in [49] that contains 26925 text lines. BLSTM based Urdu Nastaleeq recognition system was introduced for the first time by Ul-Hassan et al. [48], who suggested in conclusion that MDLSTM can perform more accurately. Later on, Naz et al. [49, 50] devised two different ways of statistical feature based MDLSTM and achieved accuracy of 94.77% and 96.04%. All of presented works except [49, 50, 52, 53], used raw pixel values as features.

To best of my knowledge, no prior LSTM based study is presented on ligature based 。 In this study, we use the same number of sentences for training i.e. 6800, as used in [29, 49, 50, 52, 53]. The systems [29, 49-53] recognize 43 classes of characters with accuracy ranges between 88.94% and 98.12%. Ul-Hassan et al. [48] used 191 and 91 classes for recognition and achieved 86.4% and 94.8% accuracies, respectively. The proposed ligature based GBLSTM system uses raw pixel values as input, with the same number of training and testing sentences as used in [29, 49, 50, 52, 53] and recognizes 3604 ligature classes with an accuracy of 96.71%. Thus, element-wise production of forward and backward layers produces very successful results by using BLSTM. The accuracy of the system is better than LSTM based systems as it learned maximum number of (ligature) classes. Furthermore, results of single layered GBLSTM are comparable with already proposed MDLSTM and results can further be improved by using stacked GBLSTM.

Table 5.3
Performance of LSTM Based Urdu Nastaleeq Recognition Systems using UPTI Dataset

Study	Features	Model	Training and validation Sentences	Testing sentences	Labeling	Alignment Method	Classes	Accuracy
Ul-Hassan et al. [48]	Raw Pixel Values	BLSTM	4607 and 3405	2003	Character Based	Sliding window	Shaped=191 Unshaped=91	86.4% (shaped) 94.8% (unshaped)
Ahmed et al.[51]	Raw Pixel Values	BLSTM	12415 and 11944	2836	Character Based	Sliding window	43	88.94
Naz et al. [49]	Statistical	MDLSTM	6800 and 1600	1600	Character Based	Sliding window	43	94.77%
Naz et al. [50]	Statistical	MDLSTM	6800 and 1600	1600	Character Based	Sliding window	43	96.40%
Naz et al. [52]	Zoning	MDLSTM	6800 and 1600	1600	Character Based	Sliding window	43	93.38%
Naz et al. [29]	Raw Pixel Values	MDLSTM	6800 and 1600	1600	Character Based	Sliding window	43	98%
Naz et al. [53]	Convolutional	MDLSTM	6800 and 1600	1600	Character Based	Sliding window	43	98 ± 0.25%
This study	**Raw Pixel Values**	**GBLSTM**	6800 and 1600	1600	**Ligature Based**	**Aligned input**	**3604**	96.71%

5.6.2 Comparison of GBLSTM with Prevalent Ligature Based Urdu Recognition Systems

The GBLSTM system recognizes sentences on the basis of ligatures. Therefore, a comparison of ligature based recognition systems is presented in Table 5.4. It is obvious that GBLSTM outperforms all prevalent ligature based recognition systems. Although, the accuracy of Lehal and Rana [31] is 98%, but the authors used a very small dataset for training and testing purpose, that may result in overfitting.

5.6.3 Number of Ligature Classes Incorrectly Predicted by GBLSTM

It is very interesting that majority of ligatures classes are misclassified less than 10 times. As per best results of GBLSTM, only 11 out of 3604 classes of ligatures are misclassified

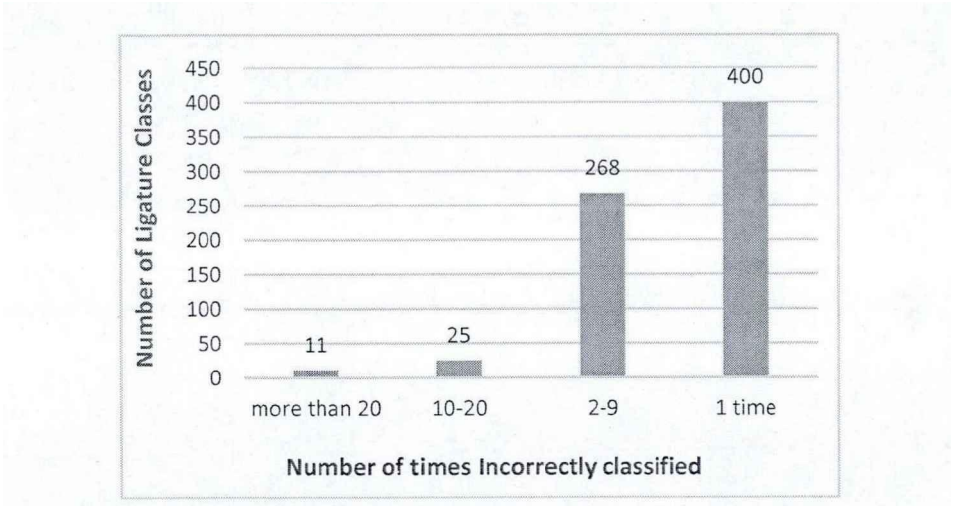


Fig. 5.11 Number of times incorrect ligature class predictions

Table 5.4
Performance of Ligature based Urdu Nastaleeq Recognition Systems

Study	Features	Model	Train and Test set ligatures	Accuracy
Javed et al. [15]	Discrete Cosine Transforms	HMM	1282 and 3655	92%
Javed and Hussain [47]	Discrete Cosine Transforms	HMM	1692	92.73%
Ahmad et al. [119]	Stacked Denoising Autoencoders	Softmax	178573	96%
Sabbour and Shafait [59]	Shape context	KNN	10000	91%
Lehal and Rana [31]	Discrete Cosine Transforms, Gabor Filters and Zoning	KNN, HMM, SVM	9262	98%
El-Korashy and Shafait[120]	Shape context	KNN	20000 and 18000	78.70%
This study	**Raw Pixel Values**	**GBLSTM**	**127180 and 29935**	96.71%

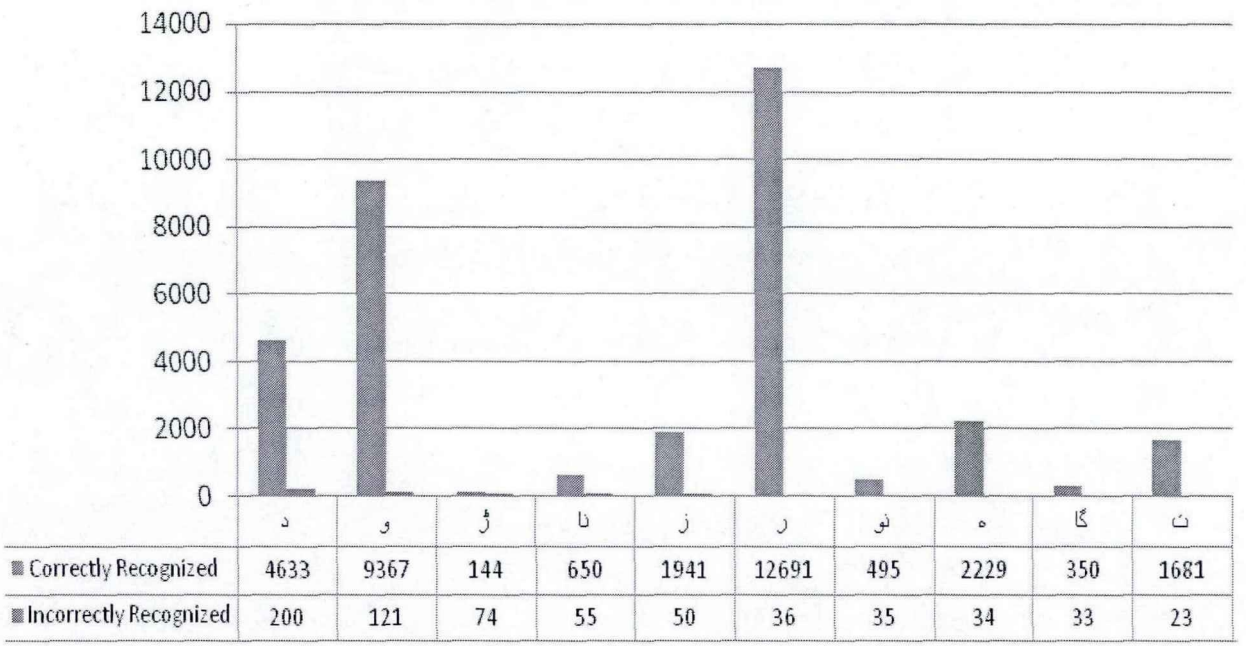


Fig. 5.12 More than 20 times incorrectly recognized ligatures for 80x80 input dimensions

more than 20 times, as shown in Fig. 5.11.

Similarly, 10 to 20 times misclassified ligature classes are 25. Misclassification in the range of 2 to 9 has 268 classes. Misclassification for one time has 400 classes that are the maximum number of ligature classes misclassified during testing.

5.6.4 Top 10 Misclassified Classes

Figure 12 shows that seven out of top ten misclassified ligature classes are characters that are mostly written smaller in size. Remaining three classes are bigram ligatures. The character "د" is at the top of list with 200 numbers of misclassifications; however, it is classified correctly with 95.86% of accuracy.

The character "و" stands at 2nd position in list with 121 number of misclassifications; however, it is classified correctly with 98.72% accuracy. Although the character "ڑ" is on third position in the list, but it has the least accuracy of 66.1% out of all ligature classes. The bigram "لگ" is next frequent misclassified ligature class with minimum accuracy of 91.38%.

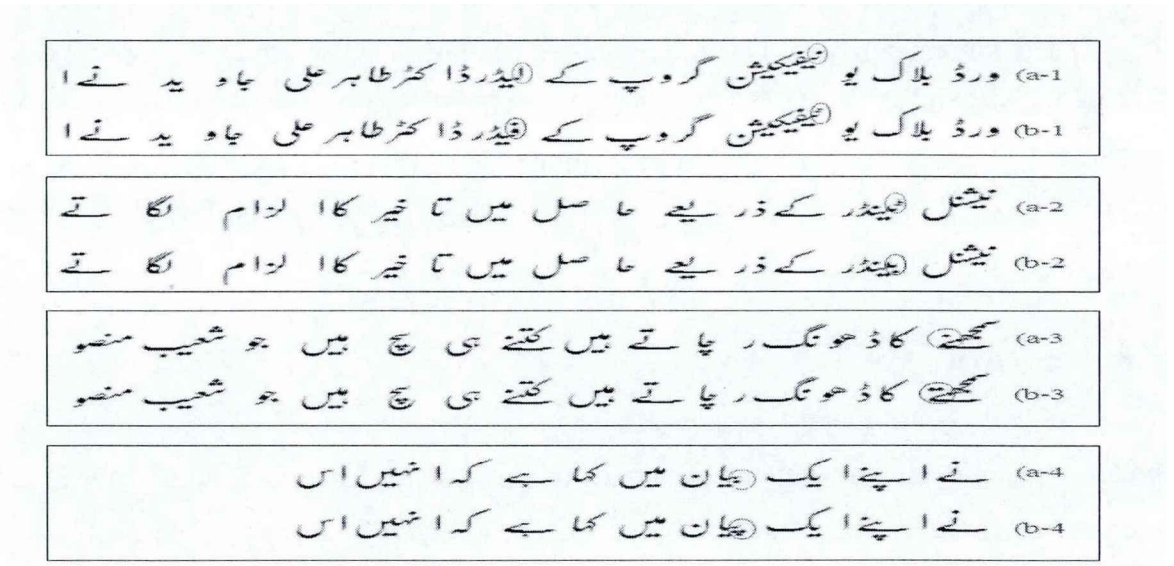


Fig. 5.13 Input and predicted sentences by the proposed GBLSTM recognition system: a-1, a-2, a-3 and a-4 represent the input sentences, while b-1, b-2, b-3, and b-4 represent the predicted outputs. Red circles point out the incorrectly predicted parts for the corresponding blue encircled parts of sentence.

5.6.5 Performance Analysis on the Basis of Full Predicted Sentences

Fig. 5.13 illustrates 4 input text line images a-1, a-2, a3, and a-4 that are fed to the recognition system. The predicted sequences of ligatures are shown as text lines b-1, b-2, b-3 and b-4 with red circles pointing to prediction error in the same figure. In all 4 sentences, primary ligatures are correctly predicted, but with incorrect dot/diacritics, except 2nd error of first sentence. The predicted primary ligature in 2nd error of first sentence (from right to left) is almost similar to the original one with only difference of starting character.

5.7 Conclusion

In this chapter, a ligature based GBLSTM model for Urdu Nastaleeq text recognition is presented dissimilar to character based LSTM strategies defined in earlier contributions. The proposed system takes segmented sentences labelled with aligned ligatures. The proposed model works on the basis of gated contextual information among ligatures of a sentence, and it recognizes sentences with 96.71% accuracy. Also, the proposed model neither needs to extract features nor does it make use of any window size or CTC layer.

The maximum numbers of character classes recognized in the referenced work [48] is 191 with the accuracy of 86.4%. Also, prevalent MDLSTM based contributions [29, 49-53] used 43 classes of characters for recognition. The proposed single layer GBLSTM model recognizes 3604 ligature classes with an accuracy of 96.71% on the same UPTI dataset.

5.8 Summary

The underlying chapter proposed a novel ligature based BLSTM approach for sentence recognition. The prevalent contribution for Urdu sentence recognition have used only character based methods. Such systems contain very difficult process of labeling of sentences with appropriate shaped characters. On the other hand, labeling sentences with ligatures is comparatively easier. Further, GBLSTM methodology provided improved result because of combining information of forward and backward layers appropriately with the help of multiplicative gate layer. The performance of GBLSTM is comparable with the earlier contributions using multidimensional LSTM architectures.

Chapter 6

Conclusions and Potential Future Directions

Conventional machine learning methods are less appreciated with the advent of deep learning methodologies as latter produces superior results. During current epoch, similar to many other pattern recognition and machine learning domains, absolute Urdu text recognition systems are the need of the time. The more accurate and generalized Urdu language OCR could uplift the computerization and make it possible for a common Urdu speaker to take advantage of technology advancements in native language. Urdu language OCR community is keen to find effective measures for maximizing the use of OCR research in daily life, for this CLE already launched an Urdu language OCR publically. Albeit of nuanced Urdu OCR research works have been put forward by the scholastic experts, but unlike Chinese, Latin, Japanese and other languages, Urdu research is lagging behind due to lack of researchers, resources, support of governments, and organizations as well. To undo such convention, contemporary measures are presented in this thesis for the uplift of Urdu OCR. This chapter is intended to briefly present the research findings of the dissertations and subsequently to propose potential future directions.

6.1 Research Insights

The brief insights of the underlying dissertation are enlisted as follows:

First chapter of this dissertation provides the introduction of Urdu OCR systems. Further aspects include background of OCR, types of general OCR. Next, a brief discussion about complexities of Urdu Nastaleeq script. Subsequently, prevailing Urdu Nastaleeq text segmentation techniques are discussed. Finally, the motivations and contributions of dissertation are highlighted.

Second chapter provides the detailed overview of preliminaries, data sets and the organizations working on Urdu OCR. First part includes projection profile method and its types. Next, different activation functions like tanh function, with logistic regression and softmax classifiers are discussed. Next, Artificial Neural Network, autoencoder with its variants are detailed. Further, sequence learning techniques are discussed containing

hidden morkov model, recurrent neural network, long short term memory and bidirectional Long Short Term Memories as sequence learning techniques especially LSTM based are very much in focus for OCR arena. Finally, brief discussion is given about commonly used Urdu datasets and organizations working for uplift of Urdu OCRs.

In chapter 3, two effective line and ligature segmentation algorithm are presented for Nastaleeq Urdu document images. For this intent, the projection profile method is augmented with a novel Curved Split Line algorithm for segmentation of text lines of Urdu document images. Further, a comprehensive ligature segmentation algorithm is put forward and successfully segmented text lines of UPTI dataset with an accuracy of 99.08%. Overall, both proposed segmentation algorithms performed comparable with the state of art segmentation algorithms for Urdu document images.

Limitations and recommendations: Segmenting lines and ligature of a cursive language like Urdu is very hard. The proposed segmentation algorithms still need to be improved. Currently, one column document images are tested. So, the line segmentation algorithm can be improved to segment lines from multiple column document images. Similarly, text and non-text area can be demarcated with the help of improved line segmentation algorithm.

Chapter 4 proposed a novel mechanism for ligature recognition while extracting features by stacked denoising autoencoder with softmax classifier. For this purpose, features are extracted using different SDA networks and best network is chosen for testing purpose. This system recognized more than 3732 classes of ligatures with accuracy of 96%. Same data is also tested on SVM. Overall, the proposed URL-SDA learning models outperformed the SVM and prevailing models for Urdu OCRs.

Limitations and recommendations: Recognition of ligatures by SDA network with softmax at output layer proved very successful. Further, instead of SDA other deep learning approaches like Deep Convolutional Networks, Deep Belief Networks, Restricted Boltzmann Machines can also be used to Urdu document image recognition.

Chapter 5. The intuition of this chapter is to use the context aware methodology for Urdu sentence recognition. The LSTM based networks used successfully for sentence recognition. This chapter proposed a novel gated BLSTM approach for sentence recognition. The proposed gated BLSTM used ligature labelling while the earlier LSTM

based contributions used character labelling. Overall, results of proposed BLSTM system are comparable with multidimensional LSTM models.

Limitations and recommendations: For better results, multidimensional GBLSTM network can be used. Also, instead of using raw pixel values, features from deep learning method would certainly improve the recognition accuracies as discussed in Chapter 4.

6.2 Potential Future Recommendations

Urdu OCRs have been steadily emerging with more research work from last two decades. As, Urdu OCR cannot use the same techniques and methodologies as used for other non-cursive languages like Latin. One of the major obstacle for a successful Urdu OCR is the non-availability of public datasets. Different researchers are using dissimilar datasets that makes it hard to compare the performance of different contributions justly. Some potential future works are listed as follows:

• Appropriate publically available datasets:

Urdu OCR domain is still demanding the attention of all researchers for contributions towards publically available datasets.

• Handwritten Urdu OCR:

The research on recognition of Urdu handwritten document images is still in infancy stage and need attention.

• Deep learning method would certainly improve the recognition accuracies

• Collaborative work is needed among the Urdu OCR research community for a successful and usable Urdu OCR.

Bibliography

[1] H. F. Schantz, "The history of OCR: optical character recognition, Recognition Technologies Users Association,1982.

[2] La macchina che legge e che scrive (PDF), in La scienza per tutti, Year XXIII, n° 11, Milano, Casa Editrice Sozogno, 1° June 1916, p. 166. (italian)

[3] "History of Computers and Computing, Birth of the modern computer, The bases of digital computers, OCR". history-computer.com. Retrieved 2016-09-09.

[4] "Optical character recognition - History". ABBYY Technology. Retrieved 18 September 2016.

[5] J. Scott Hauger, Reading Machines for the Blind (PDF), Blacksburg, Virginia, Faculty of the Virginia Polytechnic Institute and State University, April 1995, pp. I-II, 11-13.

[6] Adnan Ul-Hasan, Generic Text Recognition using Long Short-Term Memory Networks, (Ph.D. Thesis) Department of Computer Science, University of Kaiserslautern, Germany, 2016.

[7] Shah M., Jethava B.J., "A literature review on hand written character recognition", Indian Streams Research Journal, Vol -3 , ISSUE –2, March.2013, ISSN:-2230-7850

[8] R. J. Ramteke, Imran Khan Pathan, S. C. Mehrotra, "Skew Angle Estimation of Urdu Document Images: A Moments Based Approach", International Journal of Machine Learning and Computing, Vol.1, No. 1, April 2011

[9] Muhammad Imran Razzak, Online Urdu Character Recognition In Unconstrained Enviornment, (Ph.D. Thesis) Department of Computer Science, Faculty Of Basic And Applied Sciences, International Islamic University, Islamabad, 2010.

[10] G. Weber, "Top languages," The World's, vol. 10, 2008.

[11] S. Hussain, S. Ali, and Q. Akram, "Nastalique segmentation-based approach for urdu ocr," International Journal on Document Analysis and Recognition (IJDAR), vol. 18, no. 4, pp. 357–374, 2015.

[12] Hussain, S. Complexity of Asian Writing Systems: A Case Study of Nafees Nasta'leeq for Urdu

[13] A. Daud, W. Khan, and D. Che, "Urdu language processing: a survey,"Artificial Intel ligence Review, pp. 1–33, 2016.

[14] S. Naz, K. Hayat, M. I. Razzak, M. W. Anwar, S. A. Madani, and S. U.Khan, "The optical character recognition of urdu-like cursive scripts," Pattern Recognition, vol. 47, no. 3, pp. 1229–1248, 2014.

[15] S. T. Javed, S. Hussain, A. Maqbool, S. Asloob, S. Jamil, and H. Moin, "Segmentation free Nastalique Urdu ocr," World Academy of Science, Engineering and Technology, vol. 46, pp. 456–461, 2010.

[16] S. A. Husain, "A multi-tier holistic approach for Urdu Nastaleeq recognition," in Multi topic conference, 2002. Abstracts. INMIC 2002. International, pp. 84–84, IEEE, 2002.

[17] Wali, A. and Hussain, S. "Context Sensitive Shape-Substitution in Nastaliq Writing system: Analysis and Formulation", in the Proceedings of International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering (CISSE), 2006.

[18] Danish Altaf Satti, Offline Urdu Nastaliq OCR for Printed Text using Analytical Approach, (Ph.D. Thesis) Department of Computer Science, Quaid-i-Azam University, Islamabad, Pakistan, January, 2013.

[19] A. Muaz, Urdu optical character recognition system (Master's thesis). National University of Computer & Emerging Sciences Lahore, Pakistan, 2010.

[20] M. Asad, A.S. Butt, S. Chaudhry, S. Hussain, Rule-based expert system for urdu Nastaleeq justification, in: Proceedings of the 8th International Multitopic IEEE Conference (INMIC'04), 2004, pp. 591–596

[21] U. Pal, A. Sarkar, Recognition of printed Urdu script, in: Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003), 2003, pp. 1183–1187.

[22] D.B. Megherbi, S.M. Lodhi, A.J. Boulenouar, Fuzzy-logic-model-based technique with application to Urdu character recognition, Proceedings of the SPIE Applications of Artificial Neural Networks in Image Processing V 3962 (2000) 13–24.

[23] Z.A. Shah, Ligature based optical character recognition of Urdu-Nastaleeq font, in: Proceedings of the 6th International Multitopic IEEE Conference (INMIC'02), 2002, pp. 25–25.

[24] S.A. Husain, A multi-tier holistic approach for urdu Nastaliq recognition, in: Proceedings of the 6th International Multitopic IEEE Conference (INMIC'02), 2002, pp. 528–532

[25] U.R. Ahmed, Design and implementation report of optical Urdu text recognition (Master's thesis). COMSATS Institute of Information Technology, Lahore, Pakistan, 2004.

[26] S.T. Javed, Investigation into a segmentation based OCR for the Nastaleeq writing system (Master's thesis). National University of Computer & Emerging Sciences, Lahore, Pakistan, 2007

[27] S.A. Sattar, A technique for the design and implementation of an OCR for printed Nastalique text (Ph.D. dissertation). NED University of Engineering & Technology, Karachi, Pakistan, 2009.

[28] U. Iftikhar, Recognition of Urdu Ligatures (Master's thesis). VIBOT Consortium and German Research Center for Artificial Intelligence (DFKI), 2011.

[29] S. Naz, A. I. Umar, R. Ahmad, M. I. Razzak, S.F. Rashid, and F. Shafait, "Urdu Nasta'liq text recognition using implicit segmentation based on multi-dimensional long short term memory neural networks," SpringerPlus, vol. 5(1), p.2010, 2016.

[30] K. Saeed, M. Albakoor, "Region growing based segmentation algorithm for typewritten and handwritten text recognition," Applied Soft Computing, Vol. 9 issue. 2, pp.608-617, 2009.

[31] G. S. Lehal and A. Rana, "Recognition of nastalique urdu ligatures," in Proceedings of the 4th International Workshop on Multilingual OCR, p. 7, ACM, 2013.

[32] Tesseract, June 2014. http://code.google.com/p/tesseract-ocr/.

[33] G. Tauscheck. Reading machine, 12 1929

[34] W. J. Hannan. R. C. A. Multifont Reading Machine. Optical Character Recognition, pages 3–14, 1962.

[35] ERA. An Electronic Reading Automation. Electronics Engineering, pages 189–190, 1957.

[36] T. Iijima, Y. Okumura, and K. Kuwabara. New Process of Character Recognition using Sieving Method. Information and Control Research, 1(1):30– 35, 1963.

[37] Y. Mei, X. Wang, and J. Wang. An Effient Character Segmentation Algorithm for Printed Chinese Documentation. In UCMA, pages 183–189, 2013.

[38] J. Cowell, F. Hussain, A fast recognition system for isolated arabic characters, in: Proceedings of the 6th International Conference on Information Visualisation, London, UK, 2002, pp. 650–654.

[39] S.M. Azam, Z.A. Mansoor, M. Sharif, On fast recognition of isolated characters by constructing character signature database, in: Proceedings of the International Conference on Emerging Technologies (ICET'06), 2006, pp. 568–575.

[40] Z. Ahmad, J.K. Orakzai, I. Shamsher, A. Adnan, Urdu Nastaleeq optical character recognition, in: Proceedings of the World Academy of Science, Engineering and Technology, vol. 26, 2007.

[41] I. Shamsher, Z. Ahmad, J.K. Orakzai, A. Adnan, OCR for printed urdu script using feed forward neural network, Proceedings of the World Academy of Science, Engineering and Technology 23 (2007) 172–175.

[42] T. Nawaz, S.A.H.S. Naqvi, H. ur Rehman, A. Faiz, Optical character recognition system for Urdu (Naskh font) using pattern matching technique, International Journal of Image Processing (IJIP) 3 (3) (2009) 92–104.

[43] J. Tariq, U. Nauman, M.U. Naru, Softconverter: a novel approach to construct OCR for printed urdu isolated characters, in: Proceedings of the 2nd International Conference on Computer Engineering and Technology (ICCET'10), vol. 3, Singapore, 2010, pp. V3–495–V3–498.

[44] S. Zaman, W. Slany, F. Sahito, Recognition of segmented Arabic/Urdu characters using pixel values as their features, in: Proceedings of the 1st International Conference on Computer and Information Technology (ICCIT'2012), 2012, pp. 507–512.

[45] I.K. Pathan, A. Ahmed. A. Ali, R.J. Ramteke, "Recognition of offline handwritten isolated Urdu character ", International Journal on Advances in Computational Research, Vol. 4, Issue 1, pp. 117-121, 2012

[46] K. Khan, R. Ullah, N. Khan, K. Naveed, "Urdu Character Recognition using Principal Component Analysis", International Journal of Computer Applications (0975 – 8887), Volume 60– No.11, December 2012

[47] S. T. Javed, S. Hussain, "Segmentation Based Urdu Nastalique OCR," In Iberoamerican Congress on Pattern Recognition, pp. 41-49. Springer Berlin Heidelberg, 2013.

[48] A. Ul-Hasan, S. B. Ahmed, F. Rashid, F. Shafait, and T. M. Breuel, "Offline printed urdu nastaleeq script recognition with bidirectional lstm networks," in Document Analysis

and Recognition (ICDAR), 2013 12th International Conference on, pp. 1061 –1065, IEEE, 2013.

[49] S. Naz, A. I. Umar, R. Ahmad, S. B. Ahmed, S. H. Shirazi, and M. I. Razzak, "Urdu nastaliq text recognition system based on multidimensional recurrent neural network and statistical features," Neural Computing and Applications, pp. 1–13, 2015.

[50] 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, 2015.

[51] 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, pp. 1–11, 2015.

[52] S. Naz, S. B. Ahmed, R. Ahmad, & M. I. Razzak, "Zoning features and 2DLSTM for Urdu text-line recognition," Procedia Computer Science, Vol. 96, pp. 16-22, 2016.

[53] S. Naz, A. I. Umar, R. Ahmad, I. Siddiqi, S. B. Ahmed, M. I. Razzak, and F. Shafait, "Urdu Nastaliq recognition using convolutional recursive deep learning," Neurocomputing, vol. 243, pp.80-87, 2017.

[54] S.T. Javed, S. Hussain, Improving Nastalique-specific pre-recognition process for Urdu OCR, in: Proceedings of the 13th International Multitopic IEEE Conference (INMIC'09), pp. 1–6, 2009.

[55] S.A. Sattar, S. Haque, M.K. Pathan, Q. Gee, Implementation challenges for nastaliq character recognition, in: Wireless Networks, Information Processing and Systems, ser. Communications in Computer and Information Science, vol. 20. Springer, Berlin, Heidelberg, 2009, pp. 279–285.

[56] S.A. Sattar, S. ul Haq, M.K. Pathan, A finite state model for urdu nastalique optical character recognition, International Journal of Computer Science and Network Security (IJCSNS) 9 (9) (2009).

[57] H. Malik, M.A. Fahiem, Segmentation of printed urdu scripts using structural features, in: Proceedings of the 2nd International Conference in Visualisation (VIZ'09), pp. 191–195, 2009.

[58] M. Akram, S. Hussain, Word segmentation for urdu OCR system, in: Proceedings of the 8th Workshop on Asian Language Resources. Asian Federation for Natural Language Processing, Beijing, China, pp. 87–93, 2010.

[59] N. Sabbour, F. Shafait. "A segmentation-free approach to Arabic and Urdu OCR", SPIE Document Recognition and Retrieval XX, DRR'13, San Francisco, CA, USA, Feb 2013.

[60] Q. Akram, S. Hussain, F. Adeeba, S. Rehman, and M. Saeed, "Framework of Urdu Nastalique Optical Character Recognition System", in the Proceedings of Conference on Language and Technology 2014 (CLT 14), Karachi, Pakistan. (URL: http://cs.dsu.edu.pk/clt14/).

[61] Akram, A., Hussain S., Habib Z., "Font Size Independent OCR for Noori Nastaleeq" , Proceedings of Graduate Colloquium on Computer Sciences (GCCS), Department of Computer Science, FAST-NU Lahore, Volume 1, 2010

[62] D. A. Satti and K. Saleem, "Complexities and implementation challenges in offline Urdu nastaliq OCR," in Proceedings of the Conference onLanguage & Technology, pp. 85–91, 2012.

[63] H. Sarfraz, A. Dilawari, and S. Hussain, "Assessing Urdu language support on the multilingual web," in Proceedings of the 12th AMIC Annual Conference on e-Worlds: governments, Business and Civil Society, Asian Media Information Center, Singapore, 2003.

[64] M. Javed, P. Nagabhushan, B.B. Chaudhuri, "Extraction of Projection Profile, Run-Histogram and Entropy Features Straight from Run-Length Compressed Text-Documents," In Pattern Recognition (ACPR), 2nd IAPR Asian Conference on (pp. 813-817), IEEE, November 2013.

[65] M. Arivazhagan, H. Srinivasan, and S. Srihari, "A statistical approach to line segmentation in handwritten documents," Proc. SPIE 6500, Document Recognition and Retrieval XIV, 65000T, January 2007.

[66] R. Kasturi, L. O. Gorman, and V. Govindaraju, "Document image analysis: A primer," Sadhana Part 1, vol. 27, pp. 3–22, 2002.

[67] H. Baird, "Skew angle of printed documents," Proceedings of SPSE's 40th Annual Conference and Symposium on Hybrid Imaging Systems,pp. 21–24, 1987.

[68] G. Nagy, S. Seth, and M. Viswanathan, "A prototype document image analysis system for technical journals," Computer, vol. 25, no. 7, pp. 10–22, 1992.

[69] L. Likforman-Sulem, A. Zahour, and B. Taconet, "Text line segmentation of historical documents: a survey," International Journal of Document Analysis and Recognition (IJDAR), vol. 9, pp. 123–138, April 2007.

[70] Y. Bengio, Learning deep architectures for AI, Foundations and Trends in Machine Learning 1(2) pages 1-127.

[71] P. Vincent, H. Larochelle, Y. Bengio, and P.A. Manzagol, "Extracting and composing robust features with denoising autoencoders," In Proceedings of the 25th international conference on Machine learning, pp. 1096-1103, Jul, 2008.

[72] G.E. Hinton and R.R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science. 313(5786), pp.504-507, Jul,2006.

[73] P. Vincent,H. Larochelle, I. Lajoie, Y. Bengio Y, and P.A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," The Journal of Machine Learning Research, vol. 11, no. Dec,pp.3371-3408, Mar, 2010

[74] H. Larochelle, D. Erhan, and P. Vincent, "Deep learning using robust interdependent codes," In International Conference on Artificial Intelligence and Statistics, pp. 312-319,2009.

[75] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," Advances in neural information processing systems, vol. 19, pp.153, Dec, 2007.

[76] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," Unsupervised and Transfer Learning Challenges in Machine Learning, vol. 7, no. 1, pp. 37-50, 2012.

[77] L.R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," Proceedings of the IEEE, 77(2), pp.257-286, 1989.

[78] Jayaraman; et al. Digital Image Processing. Tata McGraw Hill Education, ISBN 9781259081439,2009.

[79] Rosin, Paul; Collomosse, John (2012). Image and Video-Based Artistic Stylisation. Springer. p. 92. ISBN 9781447145196.

[80] Marques, Oge (2011). Practical Image and Video Processing Using MATLAB. Wiley. pp. 275–276. ISBN 9781118093481.

[81] A. K. Jain, M. N. Myrthy, and P. J. Flynn. Data clustering: A survey. ACM Computing Survey, 31 (3):264--323, 1 999.

[82] Jaekyu Ha, R.M. Haralick, I.T. Phillips, Recursive X-Y cut using bounding boxes of connected components, International Conference on Document Analysis and Recognition, ICDAR 1995.

[83] J. L. Meunier, "Optimized xy-cut for determining a page reading order," In Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on (pp. 347-351). IEEE, August, 2005.

[84] K. Y. Wong, R. G. Casey, and F. M. Wahl, "Document analysis system," IBM Journal of Research and Development, vol. 26, no. 6, pp. 647–656, 1982

[85] F. Shafait, D. Keysers, T. Breuel, "Performance evaluation and benchmarking of six-page segmentation algorithms," IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(6), 941-954, 2008.

[86] A. L. Maas, A. Y. Hannun, & A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," In Proc. ICML, Vol. 30, No. 1, June, 2013.

[87] D. Kriesel, A Brief Introduction to Neural Networks, framework of a seminar of the University of Bonn in Germany, published online under www.dkriesel.com on 5/27/2005.

[88] Š. Raudys, "Evolution and generalization of a single neurone: I. single-layer perceptron as seven statistical classifiers," Neural Networks, 11(2), 283-296, 1998.

[89] S. Haykin, N. Network, A comprehensive foundation. Neural Networks, 2 (2004).

[90] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, et al., "Improvements to Platt's SMO algorithm for SVM classifier design," Neural computation, Vol. 13(3), pp.637-649, 2001.

[91] A. Graves, A. R. Mohamed, & G. Hinton, "Speech recognition with deep recurrent neural networks," In Acoustics, speech and signal processing (icassp), 2013 ieee international conference on (pp. 6645-6649). IEEE, May,2013.

[92] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, S. Khudanpur, "Recurrent neural network based language model," In Interspeech(Vol. 2, p. 3) , September, 2010.

[93] Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin, "A neural probabilistic language model," Journal of machine learning research, 3(2), 1137-1155, Feb, 2003.

[94] M. Auli, M. Galley, C. Quirk, G. Zweig, "Joint Language and Translation Modeling with Recurrent Neural Networks," In EMNLP (Vol. 3, No. 8, p. 0). October, 2013.

[95] J., Mao, W., Xu, Y., Yang, J., Wang, Z., Huang, A. Yuille, "Deep captioning with multimodal recurrent neural networks," arXiv preprint arXiv:1412.6632, 2014.

[96] Y. Bengio, P. Simard, P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," IEEE Transactions on Neural Networks, 5(2), 157–166, 1994.

[97] S. Hochreiter, J. Schmidhuber, "Long short-term memory," Neural computation, 9(8), 1735-1780, 1997.

[98] Y. Shen, J. Chen, & X. Huang, "Bidirectional Long Short-Term Memory with Gated Relevance Network for Paraphrase Identification," In International Conference on Computer Processing of Oriental Languages (pp. 39-50), Springer International Publishing, December, 2016.

[99] M.W. Sagheer, C.L. He, N. Nobile, C.Y. Suen, A new large Urdu database for off-Line handwriting recognition, 5716 (2009) 538–546

[100] A. Raza, I. Siddiqi, A. Abidi, F. Arif, An unconstrained benchmark Urdu handwritten sentence database with automatic line segmentation, in: Proceedings of the International Conference on Frontiers in Handwriting Recognition (ICFHR'12), 2012, pp. 491–496.

[101] K. S. Kumar, S. Kumar, and C. Jawahar, "On segmentation of documents in complex scripts," in Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on, vol. 2, pp. 1243–1247, IEEE, 2007.

[102] T. M. Breuel, "Two geometric algorithms for layout analysis," in International workshop on document analysis systems, pp. 188–199, Springer,2002.

[103] F. Shafait, A. Hasan, D. Keysers, and T. M. Breuel, "Layout analysis of Urdu document images," in 10th IEEE Int. Multitopic Conference, INMIC'06, Islamabad, Pakistan, Dec. 2006

[104] S. S. Bukhari, F. Shafait, and T. M. Breuel, "High performance layout analysis of arabic and Urdu document images," in Document Analysis and Recognition (ICDAR), 2011 International Conference on, pp. 1275–1279, IEEE, 2011.

[105] S. S. Bukhari, F. Shafait, and T. M. Breuel, "Towards generic text-line extraction," in Document Analysis and Recognition (ICDAR), 2013 12th International Conference on, pp. 748–752, IEEE, 2013.

[106] S. S. Bukhari, F. Shafait, and T. M. Breuel, "Text-line extraction using a convolution of isotropic gaussian filter with a set of line filters," in Document Analysis and Recognition (ICDAR), pp. 579–583, IEEE, 2011.

[107] S. S. Bukhari, T. M. Breuel, and F. Shafait, "Textline information extraction from grayscale camera-captured document images," in 16th of IEEE International Conference on Image Processing (ICIP), pp. 2013–2016, IEEE, 2009.

[108] S. T. Javed and S. Hussain, "Improving nastalique specific prerecognition process for Urdu ocr," in Multitopic Conference, 2009. INMIC 2009. IEEE 13th International, pp. 1–6, IEEE, 2009.

[109] G. S. Lehal, "Ligature segmentation for Urdu ocr," in Document Analysis and Recognition (ICDAR), 2013 12th International Conference on,pp. 1130–1134, IEEE, 2013.

[110] I. U. Din, Z. Malik, I. Siddiqi, and S. Khalid, "Line and ligature segmentation in printed Urdu document images," J. Appl. Environ. Biol. Sci, vol. 6, no. 3S, pp. 114–120, 2016.

[111] H. S. Baird, "Document image defect models," in Structured Document Image Analysis, pp. 546–556, Springer, 1992.

[112] A. Pal and J.D. Pawar, "Recognition of online handwritten Bangla characters using hierarchical system with Denoising Autoencoders," In International Conference on Computation of Power, Energy Information and Communication, pp. 47-51, Apr,2015.

[113] A. Pal, "Bengali handwritten numeric character recognition using denoising autoencoders," In IEEE International Conference on Engineering and Technology, pp. 1-6, Mar, 2015.

[114] F. Feng, X. Wang, R. Li, and I. Ahmad, "Correspondence Autoencoders for Cross-Modal Retrieval," ACM Transactions on Multimedia Computing, Communications, and Applications, vol. 12, no. 1s, pp.26, Oct, 2015.

[115] O.K. Oyedotun, E.O. Olaniyi, and A. Khashman, "Deep Learning in Character Recognition Considering Pattern Invariance Constraints," International Journal of Intelligent Systems and Applications, vol. 7, no. 7, pp.1, Jun, 2015.

[116] S. Sardar and A. Wahab, "Optical character recognition system for Urdu," In International Conference on Information and Emerging Technologies (ICIET), pp. 1-5, Jun, 2010.

[117] S. Naz, A. I. Umar, S. H. Shirazi, S. A. Khan, I. Ahmed, and A. A. Khan, "Challenges of urdu named entity recognition: A scarce resourced language," Research Journal of Applied Sciences, Engineering and Technology, vol. 8, no. 10, pp. 1272–1278, 2014.

[118] D. Hussain, D. Hassan, P. Guo, "Improved Arabic Word Classification using Spatial Pyramid Matching Method," in International Conference Image and Vision Computing New Zealand (IVCNZ, 2011).

[119] I. Ahmad, X. Wang, R. Li, and S. Rasheed, "Offline Urdu Nastaleeq optical character recognition based on stacked denoising autoencoder," China Communications, vol. 14, no. 1, pp. 146–157, 2017.

[120] A. El-Korashy and F. Shafait, "Search space reduction for holistic ligature recognition in Urdu nastalique script," in Document Analysis and Recognition (ICDAR), 2013 12th International Conference on, pp. 1125– 1129, IEEE, 2013.

[121] A. Graves, Supervised sequence labelling. Springer, 2012.

[122] J. S. Bridle, "Probabilistic interpretation of feed forward classification network outputs, with relationships to statistical pattern recognition," in Neurocomputing, pp. 227–236, Springer, 1990.

[123] S. Canyameres Masip, A. M. L´opez Pe˜na, et al., "On the use of convolutional neural networks for pedestrian detection," 2015.

[124]] A. Graves, M. Liwicki, S. Fern´andez, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 31, no. 5, pp. 855–868, 2009.

[125] D. Wu, Human Action Recognition Using Deep Probabilistic Graphical Models. PhD thesis, University of Sheffield, 2014.

[126] R. Plamondon and S. N. Srihari, "Online and off-line handwriting recognition: a comprehensive survey," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 22, no. 1, pp. 63–84, 2000

[127] H. Jaeger, "Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and theecho state network approach," GMDForschungszentrum Informationstechnik, 2002.

[128] A. Senior and T. Robinson, "Forward-backward retraining of recurrent neural networks," Advances in Neural Information Processing Systems, pp. 743–749, 1996.

[129] A. Graves, "Offline Arabic handwriting recognition with multidimensional recurrent neural networks," in Guide to OCR for Arabic Scripts, pp. 297–313, Springer, 2012.

[130] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in Advances in neural information processing systems, pp. 545–552, 2009

[131] A. Graves, S. Fern´andez, and J. Schmidhuber, "Bidirectional lstm networks for improved phoneme classification and recognition," in International Conference on Artificial Neural Networks, pp. 799–804, Springer, 2005.

[132] A. Graves, M. Liwicki, H. Bunke, J. Schmidhuber, and S. Fern´andez, "Unconstrained on-line handwriting recognition with recurrent neural networks," in Advances in Neural Information Processing Systems, pp. 577–584, 2008.

[133] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," Signal Processing, IEEE Transactions on, vol. 45, no. 11, pp. 2673–2681, 1997.

[134] Z. Wojna, Fast methods in training deep neural networks for image recognition. PhD thesis, University College London, 2015.

[135] Z. Lian, X. Jing, X. Wang, H. Huang, Y. Tan, and Y. Cui, "Dropconnect regularization method with sparsity constraint for neural networks," Chinese Journal of Electronics, vol. 25, no. 1, pp. 152–158, 2016.

[136] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11) (G. J. Gordon and D. B. Dunson, eds.), vol. 15, pp. 315–323, Journal of Machine Learning Research - Workshop and Conference Proceedings, 2011.

[137] I. Ahmad, X. Wang, R. Li, M. Ahmad, and R. Ullah, "Line and Ligature Segmentation of Urdu Nastaleeq Text," in IEEE Access, pp. 1-17, 2017, DOI: 10.1109/ACCESS.2017.2703155.

[138] A. Graves, S. Fern´andez, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in Proceedings of the 23rd international conference on Machine learning, pp. 369–376, ACM, 2006.

Acknowledgement

First and foremost, I would like to thank Almighty Allah who gave me strength and courage to accomplish this work. I would like to present my gratitude to my noble parents who grew me up and who always keep me in their prayers. They remained the source of encouragement, guidance, throughout my life.

I would like to present my special and cordial gratitude to my kind advisor Prof. Dr. Wang Xiao Jie for his support, motivation and professional guidance. I found him really helpful, motivating and encouraging. He was not only the source of knowledge and technical studies but also very gentle human. Indeed, he is the one who accept me as part of his research group and provide me this golden opportunity to work under his kind supervision. A special gratitude goes to my respected teacher, Prof. Dr. Ruifan Li, who has enlighten me with his extensive knowledge. I would also thank other former and current lab members: Dr. Fengxiag Fang, Dr. Sue, Yuz hao Mao, Guang Liu, Wang Yue and Lei Shuyu. I also owe special thanks to my batch fellows includes Dr. Rahat Ullah khan, Muhammad Azam Zia, Hasseb Ahmad, Suhail Memon, and Muhammad Saad Khan for their critical reviews of the entire thesis and moral support. They remained as a great source of encouragement through my PhD. I wish to acknowledge my special friends includes Dr. Khalid Latif, Dr. Shahid Rasheed, Dr. Waheed ur Rahman, Dr. Manzoor Khan Achakzai, Dr. Zahid Latif, Dr. Jahangir Khan, who reviewed my thesis and gave me constructive suggestions. I am truly grateful to my brothers and sisters who are the reason of everything in my life, they always motivated me in my struggles and they are the reason behind my blessed life.

The overall support and help provided by the Center for Intelligence of Science and Technology is really appreciable, I will never forget this.

Last but not the least, I would like to thank China Scholarship Council, BUPT International Student Office, BUPT language center, BUPT graduate office and my School for all the financial and technical support provided during my stay in China.

List of Publications

Principal Author Publications

1) **Ahmad, I.**, Wang, X., Li, R., & Rasheed, S. (2017). Offline Urdu Nastaleeq optical character recognition based on stacked denoising autoencoder. China Communications, 14(1), 146-157. (SCI, IF = 0.903).

2) **Ahmad, I.,** Wang, X., Li, R., Ahmed, M., Ullah, R.(2017) Line and Ligature Segmentation of Urdu Nastaleeq Text, IEEE Access, x(x), 1-17(SCI, IF = 3.224).

3) **Ahmad, I.**, Wang, X., Yuz h. M., Liu, G., Ahmed, H., Ullah, R.(2017) Ligature based Urdu Nastaleeq sentence recognition using Gated Bidirectional Long Short Term Memory, Cluster Computing, 2017. (SCI, IF =2.09).