C02029: Doctor of Philosophy
CRICOS Code: 009469A
32903 PhD Thesis: Analytics
January 2019

A Study on
Automated Handwriting Understanding

Chandranath Adak

School of Software
Faculty of Engg. & IT
University of Technology Sydney
NSW - 2007, Australia

A Study on
Automated Handwriting Understanding

A thesis submitted in partial fulfilment of the requirements
for the degree of

Doctor of Philosophy
in
Analytics

by

Chandranath Adak

to

School of Software
Faculty of Engineering and Information Technology
University of Technology Sydney
NSW - 2007, Australia

January 2019

Abstract

Handwriting is a concatenation of graphical symbols drawn by a pen or other writing instruments, using a hand in order to represent linguistic constructs for communication and knowledge storage. These graphical marks/writing symbols have deep orthographic relation to the phonology of a spoken language. However, to a machine, handwriting is nothing but a pattern. Therefore, recognition of this pattern is performed in order to read a manuscript by a computer. Such a process of automatic character pattern recognition from an optically scanned document image is called OCR (Optical Character Recognition). Nowadays, the computer vision method is not limited to simply recognizing patterns/objects. It tries to endow the machine a human-like intelligent ability. The main goal of this research is using computer vision to bridge the gap between pattern recognition and human perception of handwriting. In this thesis, we focus on understanding the handwriting, which is beyond simply recognizing the characters by OCR. Towards this aim, we peek into the implicit information of handwriting to understand some inherent characteristics.

In this thesis, we concentrate on three aspects. First, understanding the generation of handwritten information by the writing body; second, understanding the writing strokes in regards to the quality of handwriting; third, understanding the content revealing handwritten word entities. Thus, the thesis contains three parts. Regardless of past researches on writer inspection, it is hard to find an empirical study performed on intra-variable handwriting, although such variation should be an important concern. The first part of this thesis addresses this concern. Besides, this part inspects the writer on some unconventional aspects, e.g., writing variability over struck-out texts, multiple scripts, etc. The second part makes a pioneering contribution to understanding writing stroke information in multiple facets, such as legibility, aesthetics, difficulty, and idiosyncrasy of strokes. The third part of the thesis approaches to comprehend the content of the handwritten document using computer vision, without the aid of a transcription engine or the natural language processing which, according to our knowledge, is the earliest attempt of its kind.

This research has adapted the traditional machine learning approaches as well as state-of-the-art deep learning approaches and has proposed new techniques to automate the process of handwriting understanding. The performed experiments have produced encouraging results, which ensure the applicability of the proposed research. This study has an impact on general image processing, pattern recognition, machine learning, and deep

learning domains, especially on document image processing and handwriting processing. Moreover, this research contributes to forensics for questioned document examination, biometrics for behavioral analysis through handwriting, library science/archival science for e-archiving of the manuscript, and data science. According to us, this study has pushed the frontiers of handwriting-related research.

AUTHOR'S DECLARATION

I, *Chandranath Adak* declare that this thesis, submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Software, Faculty of Engineering and Information Technology at the University of Technology Sydney, Australia, is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. This document has not been submitted for qualifications at any other academic institution. This research is supported by the Australian Government Research Training Program.

SIGNATURE: Production Note: Signature removed prior to publication.

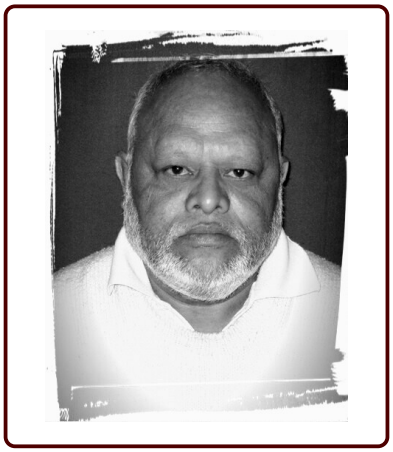
[Chandranath Adak]

DATE: 18th January, 2019

PLACE: Sydney, Australia

Dedication

To my father . . .



Late Baidyanath Adak

ACKNOWLEDGMENTS

My sincere gratitude is to my principal supervisor Prof. Michael Blumenstein, University of Technology Sydney (UTS), for his guidance and support during my Ph.D. candidature. His continuous encouragement and advice have made our research very productive. I would also like to convey my gratitude to my external co-supervisor Prof. Bidyut Baran Chaudhuri, Indian Statistical Institute (ISI), for his guidance, advice and constant motivation. For the advancement of our research, he literally acted as a critical reviewer with counter arguments and meticulous insistence on various aspects.

I would like to thank my co-supervisor Prof. Chin-Teng Lin, UTS, for his support and advice. I extend my special thanks to A/Prof. Simone Marinai, DINFO, University of Florence, Italy for his insightful advice during my visit to his lab, and IEEE Computational Intelligence Society for providing research grant for this visit.

I transferred my Ph.D. from Griffith University, Australia to UTS, following my principal supervisor Prof. Blumenstein. I heartily thank School of ICT, IIIS, and GGRS of Griffith University for supporting my initial Ph.D. research. I gratefully acknowledge all the faculty members, admin staffs, fellow researchers, GUGC student guild members of Griffith University. A special thanks to then my associate supervisor A/Prof. Jun Jo, Griffith University, for his constant support during my Griffith days.

Moreover, I express my gratitude to all the faculty members, admin staffs, fellow researchers of School of Software, CAI, and FEIT; and GRS staffs of UTS for their continuous support. I wish to thank all the faculty members and fellow researchers of CVPR Unit, Indian Statistical Institute for fruitful discussion during my ISI visits.

I also benefited from the constructive comments and suggestions made by several fellow researchers, domain experts met in conferences/academic gatherings, and also from anonymous reviewers of my papers. I thank all of them.

Heartfelt thanks to Dr. Abhijit Das (INRIA), Dr. Soumya Dutta (LANL), Tanmoy Nandi (ISI), Sumit Kr. Saha (ISI), Muhammad Saqib (UTS), Di Wu (UTS) for their help and support.

Finally, I am thankful to all my family members, especially my wife Soumi Chattopadhyay (IIITG), for constant encouragement, enthusiasm and support. I thank all those, whom I missed out from the above list.

List of Publications

Related to the Thesis :

1. C. Adak, B. B. Chaudhuri, M. Blumenstein, *An Empirical Study on Writer Identification & Verification from Intra-variable Individual Handwriting*, **IEEE Access**, vol. 7, no. 1, pp. 24738-24758, 2019.

2. C. Adak, B. B. Chaudhuri, M. Blumenstein, *A Study on Idiosyncratic Handwriting with Impact on Writer Identification*, Proc. 16th Int. Conf. on Frontiers in Handwriting Recognition (**ICFHR**), pp. 193-198, Niagara Falls, USA, 5-8 Aug., 2018.

3. C. Adak, B. B. Chaudhuri, M. Blumenstein, *Cognitive Analysis for Reading and Writing of Bengali Conjuncts*, Proc. 31st Int. Joint Conference on Neural Networks (**IJCNN**), Rio de Janeiro, Brazil, 8-13 July, 2018.

4. C. Adak, S. Marinai, B. B. Chaudhuri, M. Blumenstein, *Offline Bengali Writer Verification by PDF-CNN and Siamese Net*, Proc. 13th Int. Workshop on Document Analysis Systems (**DAS**), pp. 381-386, Austria, 24-27 Apr., 2018.

5. C. Adak, B. B. Chaudhuri, M. Blumenstein, *Legibility and Aesthetic Analysis of Handwriting*, Proc. 14th Int. Conf. on Document Analysis and Recognition (**ICDAR**), pp. 175-182, Kyoto, Japan, 9-15 Nov., 2017.

6. C. Adak, B. B. Chaudhuri, M. Blumenstein, *Impact of Struck-out Text on Writer Identification*, Proc. 30th Int. Joint Conf. on Neural Networks (**IJCNN**), pp. 1465-1471, Anchorage, Alaska, USA, 14-19 May, 2017.

7. C. Adak, B. B. Chaudhuri, M. Blumenstein, *Writer Identification by Training on One Script but Testing on Another*, Proc. 23rd Int. Conference on Pattern Recognition (**ICPR**), pp. 1148-1153, Cancun, Mexico, 4-8 Dec., 2016.

8. C. ADAK, B. B. CHAUDHURI, M. BLUMENSTEIN, *Named Entity Recognition from Unstructured Handwritten Document Images*, Proc. 12th Int. Workshop on Document Analysis Systems (**DAS**), pp.375-380, Santorini, Greece, 11-14 Apr., 2016.

OTHERS :

9. C. ADAK, B. B. CHAUDHURI, M. BLUMENSTEIN, *Offline Cursive Bengali Word Recognition using CNNs with a Recurrent Model*, Proc. 15th Int. Conf. on Frontiers in Handwriting Recognition (**ICFHR**), pp. 429-434, Shenzhen, China, 23-26 Oct., 2016.

10. C. ADAK, P. MAITRA, B. B. CHAUDHURI, M. BLUMENSTEIN, *Binarization of Old Halftone Text Documents*, Proc. **IEEE TENCON**, IEEE Conf. # 35439, pp. 1-5, Macau, China, 1-4 Nov., 2015.

11. B. B. CHAUDHURI†, C. ADAK†, *An Approach for Detecting and Cleaning of Struck-out Handwritten Text*, **Pattern Recognition**, vol. 61, pp. 282-294, January 2017. †*Joint first author*.

TABLE OF CONTENTS

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

This thesis concerns "automated handwriting understanding". Before discussing automated understanding, we briefly deliberate on handwriting and its relevance in the digital world.

Briefing on Handwriting :

"Writing" is basically a nexus of graphical symbols, used to *record* the phonology of a spoken language. This *record* is generally called as a "document", which stores and conveys information. The root of the writing, in the form of "proto-writing", is found in 7^{th} millennium BCE. It has been evolved through several civilizations of pre-history to the modern era, and portrays the "civilization's casual encephalogram (— Lance Morrow)".

Writing system : A writing system is built on the smallest meaningful contrastive unit or, basic written symbols, called as grapheme. These basic units are mainly alphabet, syllable, and logogram which form Alphabetic, Syllabic, and Logographic writing systems, respectively. Sometimes, the admixture of these basic units creates further subcategories of the writing system. The world's modern writing systems [173] are mostly classified into five categories, i.e., *Alphabetic* (e.g., Latin, Greek, etc.), *Syllabic* (e.g., Japanese Kana), *Logographic* (e.g., Chinese, Japanese Kanji, etc.), *Abugida* (or *Alphasyllabary*,

e.g., Indic, Ethiopic, etc.), and *Abjad* (or *Consonantary*, e.g., Arabic, Hebrew, etc.).

Writing body : Writing is mostly performed by human beings. Writing by some trained animal (e.g., a chimpanzee with the finger, an elephant with the trunk) can also be rarely possible. Moreover, nowadays handwriting can be performed by a (ro-)bot (e.g., AxiDraw[1]). Various human-body-parts have been used for writing, e.g., hand (finger, wrist), leg, and mouth. In this thesis, we deal with the common human-writing written using a finger grip, widely referred to as "*handwriting*".

Writing medium : The writing medium is a joint venture of the followings.
– *Tool*: These are the utensils used for writing, e.g., quill, pencil, pen (fountain, roller, gel, ball-point), etc.
– *Sheet*: On the sheet, writing is performed, e.g., parchment, papyrus, paper, etc.
– *Surface*: The sheet is placed on the surface for writing, e.g., wooden/plastic/glass board, table, etc.

Sometimes, writing is performed directly on the surface without a sheet, e.g., wall writing, linocut writing, etc. In this thesis, we focus on writing on a paper using mostly pen. An interesting circular repetition is that the writing tool started with the finger to scribble on the clay/sand surface (thousands of years ago) and now again has come back to the finger to write on the touchscreen of a smart tab.

Writing mode : On the basis of writing generation strategy and data processing, the handwritten data can be broadly categorized into two modes, i.e., *offline* and *online*. In the offline mode, after writing on a regular sheet (paper) by a usual tool (pen/pencil), the handwriting is captured by a scanner or camera and stored as a static image in the database. In the online mode, the data is captured at the time of writing using a digital pen/smart device, and contains both sequential and spatial information in the form of co-ordinate values, pen-up/down, writing pressure, speed, stroke trajectory, etc.

In this thesis, we mainly deal with offline handwriting.

Handwriting Distribution in a Document :

In today's world, a record-keeping document may be handwritten, machine-printed, or a hybrid of both. A concern of this thesis is the availability of handwriting in a document,

[1]AxiDraw: `https://axidraw.com` , retrieved on 31st Dec., 2018.

which can be of various forms, as follows.

- *Entire page :* Here, the handwriting is available throughout the page, i.e., the whole document is handwritten having no artifact. Such examples are a handwritten letter, cover letter, manuscript, etc. (refer to Figure 1.1:(a)).
- *Mixed with printed text :* In some printed documents, specific spaces are provided to be filled up by handwriting. For example, bank cheques, application forms, etc. Also, in a printed page, handwritten remarks/comments may be put in the margin. (refer to Figure 1.1:(b), (e)).
- *Only signature :* Sometimes, a document is typed/printed, and only the signature is performed by handwritten strokes. (refer to Figure 1.1:(c)).
- *Only characters :* In some printed forms, small boxed spaces are provided to fill-up each box with a single handwritten character. (refer to Figure 1.1:(d)).
- *Mixed with printed artifacts :* On a ruled-line page, handwriting can be performed where the writing may touch the ruled line. Other printed artifacts such as seal/stamp can also be present in a handwritten page and may touch the writing. (refer to Figure 1.1:(f)).
- *Mixed with hand-drawn strokes :* In a document, besides the handwriting, some hand-drawn strokes in the form of doodles, drawing, etc. can be present. (refer to Figure 1.1:(g)).

In Figure 1.1, we present some examples of handwriting distribution in the document.

Handwriting Variations :

Around 350 BCE, at the beginning of the "*De Interpretatione*", the Greek philosopher Aristotle made the following statement: "... all men have not the same writing ..." [2], which has empirically seen to be true. The handwriting intrinsically varies due to time, space (geographic location), culture, etc. Such variation among individuals may be termed as *between-writer/inter* variability. However, the handwriting of a particular individual may also vary owing to some mechanical (e.g., writing instrument, writing surface,

[2] Aristotle, "De Interpretatione", 350 BCE, Translated by E. M. Edghill, "On Interpretation", Online: `http://classics.mit.edu/Aristotle/interpretation.html`, retrieved on 31st Dec., 2018.

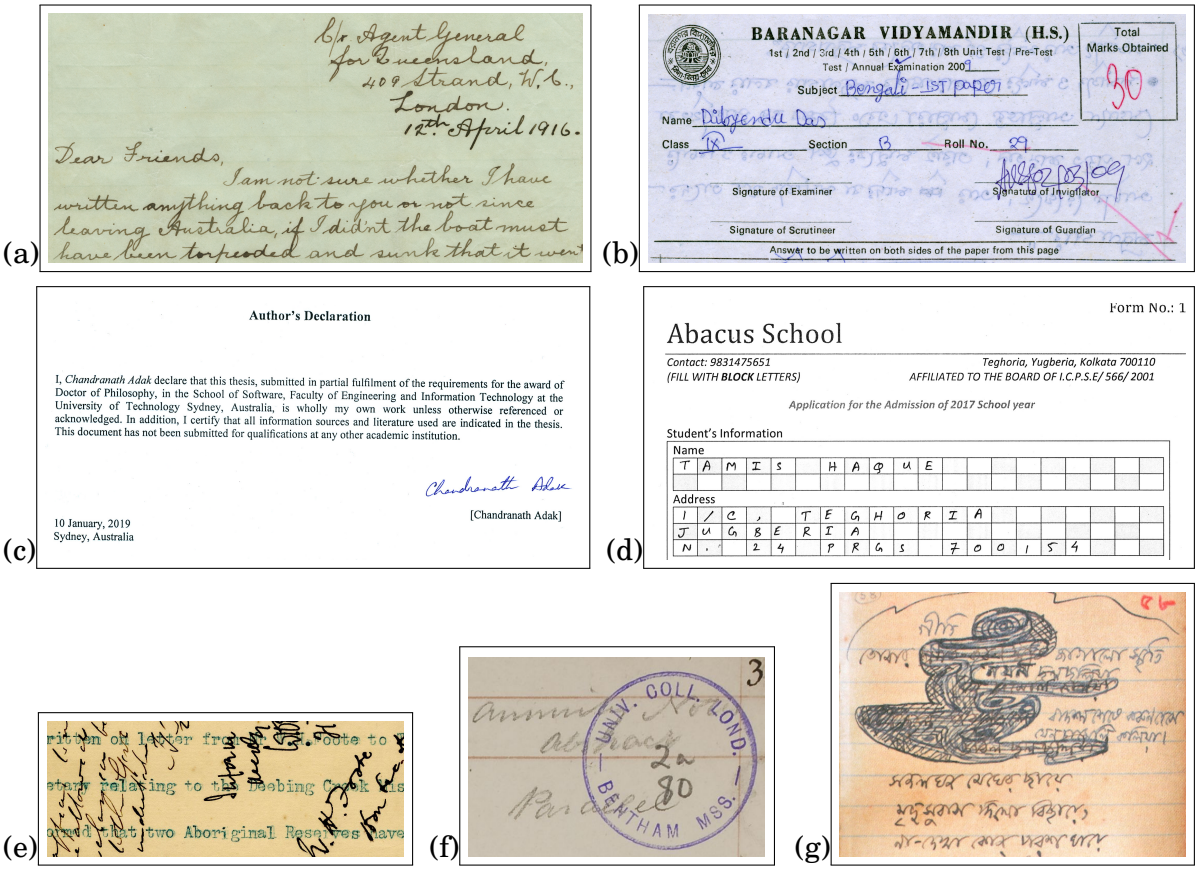


Figure 1.1: Some examples of handwriting availability in the document sample: (a) letter, (b) printed form entry, (c) signature, (d) boxed-space form entry, (e) hand-note on a typed letter, (f) writing merged with ruled-line and seal, (g) writing on a ruled-page and mixed with doodle/drawing.

etc.), physical (e.g., illness, aging, etc.), psychological (e.g., excitement, anger, mood, etc.) factors. This variation is termed as *within-writer/intra* variability. Owing to such variations, the study of handwriting becomes one of the most challenging problems.

Ideally, the inter-variability of handwriting is relatively higher than intra-variability [149, 186]. However, the intra-variability may go up due to the above-mentioned mechanical, physical, and psychological factors. On the other hand, the inter-variability may be low due to expert-level forgery and high-end modern gadgets. Such a scenario makes this study even more exciting. In Figure 1.2, we show some examples of handwriting variations. Here, Figure 1.2:(a)-(c) show inter-variable and Figure 1.2:(d)-(e) show intra-variable writings. The writing of Figure 1.2:(a) is forged by a skilled and an amateur forger separately, and are shown in Figure 1.2:(b) and Figure 1.2:(c), respectively. The handwritten samples of Figure 1.2:(d) and Figure 1.2:(e) are written by the same writer.

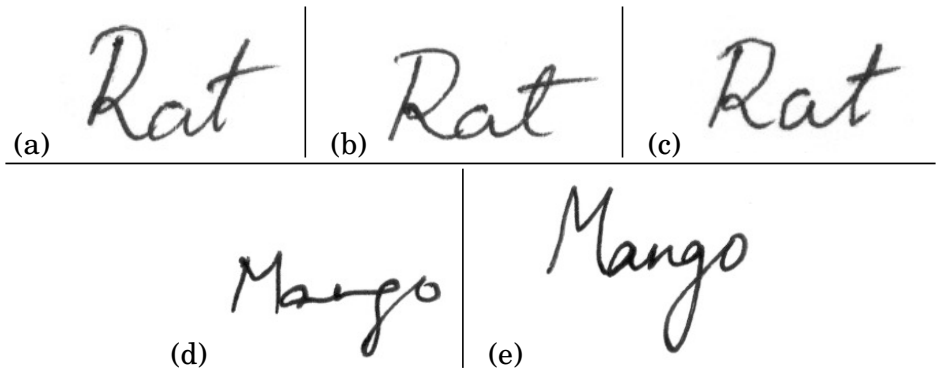


Figure 1.2: Handwriting variations: (a),(b),(c): low inter-variability although written by 3 different writers; (d),(e): high intra-variability although written by the same writer.

Handwriting Recognition :

Handwriting recognition is a task to recognize handwritten characters in order to generate a transcript of a given manuscript. Manual transcribing is a laborious job, and requires expertise and a fair amount of time with keen attention. Therefore, automation has been welcomed in this field to ease this transcript generation task. The process of automatic/computerized recognition of characters in the optically scanned document is widely called as Optical Character Recognition (OCR). The optical character recognition of handwritten characters is specifically named as OHCR (Optical Handwritten Character Recognition). This OCR/OHCR is a well-addressed problem in the literature of document image analysis [187].

However, this thesis approaches beyond the scope of the OHCR, towards the understanding of handwriting, which is discussed in the following section.

1.1 Handwriting Understanding

"Handwriting understanding" is a process to comprehend scribbled strokes for inferring some inherent knowledge. Therefore, handwriting understanding is some sort of reverse engineering may be required to peek into the generation/source information. Such source information may include the writer (writing body), writing medium (tool, sheet, surface), space (geographic location), time (date), etc. and their spatial relationship. Handwriting understanding has a broader scope compared to the limit of text reading by OCR/OHCR engine.

The essence of handwriting understanding is found around 1000 BC when few

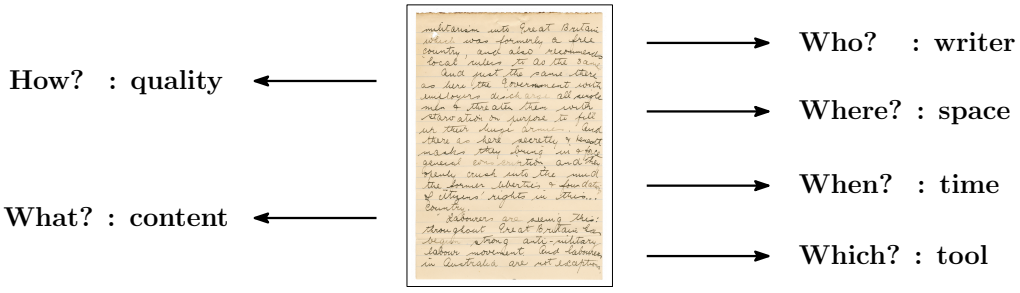


Figure 1.3: Illumination of information in handwriting.

Japanese scholars studied the bar formation in writing to judge individual characteristics [196]. During the late 20^{th} century with the booming of automation, handwriting understanding has also started to be automated. At the beginning of the 21^{st} century, the "9/11 attacks" and "2001 anthrax attacks" have escalated this automation.

The handwriting expert uses various approaches to understand a handwritten document, such as geometrical analysis of writing strokes; paper and ink testing using microscopic, infrared, chromatographic examinations, carbon dating, chemical analysis; linguistic analysis, etc. However, in this thesis, handwriting understanding is perceived as a computer vision task where an optically scanned 2D digital image is used as a raw material. Therefore, availability of the physical document, at the time of the experiment, is not necessary for this research.

Several inherent information in handwriting can be illustrated; of those few are shown in Figure 1.3. Such information can be acquired by *Wh*-questions, for example,

- "*Who* wrote the document?": to know the writer information.
- "*Where* the writing was performed?": to know the geographic location/cultural background.
- "*When* the writing was executed?": to know about the century/decade/year/date of the manuscript generation.
- "*Which* tool was used for writing?": to know the writing tool, e.g., pen, pencil, etc.
- "*How* well was the writing?": to know about the quality of writing to infer legibility, aesthetic, idiosyncrasy, etc.
- "*What* has been written?": to know the content of the writing.

To understand the writing body (writer), sometimes broader groups/demographics are considered with respect to the gender (male/female), handedness (right/left handed), age-group (child/teen/adult/old), ethnicity, etc. Such broader grouping is sometimes essential to rule-out the honest writers in order to further investigation in a smaller sample space of suspicious writers.

Clarity of handwriting stroke formation may also infer some knowledge. To this end, the legible and aesthetic appearance of writing may be explored. Besides, difficulty to scribble a writing pattern/character may comprehend the complexity of the character-stroke formation. Moreover, idiosyncratic writing strokes for character formation may influence in the understanding of some implicit information.

Sometimes, the "content", i.e., "ideas that are contained in a piece of writing"[3] is vital information to understand a handwritten document. Employing OHCR to read the entire document in order to reveal the content is a straight-forward, but a costly process. Therefore, to understand the content, the OHCR/transcription engine can possibly be bypassed.

In this thesis, our scope is limited to understanding some aspects of handwriting, such as writer, writing stroke, and content information.

Graphology :

The study of Graphology[4] is not in the scope of this thesis.

The following sections enlighten the motivation and contribution of this research work, followed by the literature review and organization of this thesis.

1.2 Motivation

Handwriting patterns are complex due to their extreme individual variations. However, handwriting can be collected with inexpensive setups and have huge scope for behavioral biometric [19]. Besides, being forensic evidence, handwriting is predominated and trusted by the legal systems of many countries world-wide. Moreover, preservation of handwritten manuscripts of notable persons and authors enrich the cultural heritage.

[3] `https://dictionary.cambridge.org/dictionary/english/content` , retrieved on 31 Dec., 2018.
[4] `https://en.oxforddictionaries.com/definition/graphology` , retrieved on 31 Dec., 2018.

The volume of digital handwritten documents has grown significantly in recent times for the purpose of e-archiving in a digital library, and provides rich streams of information. Understanding handwriting of such magnitude in real-time is becoming a stupendous challenge. Current proposals are poised at an interesting juncture, with an imminent need for effective research solutions to enhance and enrich the capabilities of automated comprehension of writing strokes. Classical proposals for handwriting understanding paradigm are primarily focused towards writer inspection on ideal documents that often neither scale to the presence of artifacts/distortions nor address the quality of writing strokes.

The primary motivation of this thesis is computerized comprehension of the underpinnings of handwritten strokes for the benefit of Forensics, Biometrics, Library and Data Sciences. In particular, this thesis has the following motivations.

- To explore several facets of an individual writing style and to inspect the writer on a more realistic/challenging handwritten document.
- To infer semantic knowledge from writing strokes/patterns.
- To extract human-understandable content from a handwritten document.

The key challenges of this research are mostly (a) inspecting a writer from a manuscript containing artifacts/struck-outs and extreme variation of writing patterns, (b) extracting implicit information from the shape of handwritten stroke, (c) revealing the content directly from a manuscript image with neither any intermediate transcript generation nor natural language processing. These challenges can be tackled by leveraging inherent characteristics of the handwriting pattern. However, in order to extract inherent features, the proposal of cutting-edge machine learning techniques is necessary, which justifies the conduct of our research.

1.3 Contributions

The main objective of this thesis is providing a human-like perception to the machine to understand handwriting. In particular, this thesis is one of the earliest attempts to comprehend some new insights from handwriting. We focus on using rigorous computer vision and machine learning algorithms to extract useful inherent semantic meaning from a pool of handwritten document images. The contributions of this thesis are threefold, i.e., understanding the writer, writing strokes, and the content, which are briefly

stated below.

Understanding the Writer :

1. *Writer verification :* Writer verification is required to check the authenticity of a writer. In this thesis, we solve this problem in a new way by feeding feature map into a convolutional neural network, which has increased the accuracy.

2. *Writer identification across scripts :* Writer identification is a process to find the exact author. In a multilingual country, people can write in multiple scripts. Therefore, it may be required to identify a writer on a particular script (e.g., English) while the available system's database contains writing in another script (e.g., Bengali). Here, we approach to tackle this problem where the writer identification is performed by training on one script and testing on another.

3. *Writer identification/verification on intra-variable handwriting :* The handwriting of an individual may vary due to some mechanical, physical, and psychological factors. Therefore, a realistic situation may arise where a particular writing type of a person is unavailable during system training, but present at the time of testing. This thesis is a pioneering attempt for writer identification/verification for highly variable individual writing styles.

4. *Writer identification from text containing strike-outs :* The past research works on writer identification mostly dealt with the ideal handwritten page without any artifact. However, a handwritten page may contain artifacts such as strike-outs. Here, we study the impact of struck-out text for writer identification.

Understanding Writing Strokes :

5. *Legibility and aesthetic analysis of handwriting :* Sometimes, understanding the clarity and beauty of writing strokes may be required, which is investigated in this thesis. From the perspective of machine learning, this contribution opens up a new area of research for providing a human-level perception to a machine.

6. *Understanding writing/reading difficulty :* An individual may face difficulties to write/read some handwritten patterns/characters. This thesis attempts to understand such difficulties, which is the earliest attempt of its kind.

7. *Idiosyncrasy understanding :* A writer may have his/her own idiosyncratic handwriting style. This idiosyncratic writing understanding may reveal some important information about individuality. We study such idiosyncrasy understanding, which is a pioneering effort.

Understanding the Content :

8. *Content understanding :* Automated content understanding is essential for managing a huge pool of documents. It is also required for information retrieval. In this thesis, we approach to comprehend the content of a handwritten document. The content analysis is performed here directly from the document image without any intermediate transcript generation, which is the earliest attempt of its kind.

1.4 Related Literature

In this section, we briefly review the literature, which includes the related works on writer investigation, handwriting stroke understanding, and content understanding, followed by existing major handwriting databases.

1.4.1 Writer Investigation

Writer investigation arises in a situation where a handwritten specimen may be claimed by an individual or unclaimed by anyone. Hence writer verification and identification need to be explored [145].

Writer verification is a task where if a person claims to be the writer of a handwritten specimen, then the system conducts one-to-one matching with his/her previously stored writing in a database to decide whether the claim is true or false. *Writer identification* works for unclaimed writing to find its writer by one-to-many comparison with the stored handwritten samples of multiple writers in a database. In other words, the writer verification system answers “yes”/“no” when asked “Is this handwriting of writer-A?”, and the writer identification system, when asked “Whose handwriting is this?”, answers the specific “writer-id”, e.g., writer-B.

The literature of the writer investigation is rich and vast. Therefore, we discuss only major past works on writer identification, writer verification, and writer demographics, followed by some interesting works on writer inspection.

1.4.1.1 Writer Identification

A comprehensive survey on writer identification, up to the year 1989, has been reported in [186]. More recent advancement in this field can be found in [252]. In this subsection, we briefly review the past works on offline automated writer identification from the viewpoint of feature extraction and classification.

Feature : For automatic writer identification, some features are extracted from the handwritten samples. Among those features, some are directly computed from the pixel-level representation of handwriting stroke pattern, commonly termed as *spatial* domain features. On the other hand, the pattern image is transformed into another domain, and the features calculated in this domain are referred to as *transform* domain features.

The typical *spatial* domain features are directional [107], hinge features [202] obtained from contour [149]/edge [146]/skeleton [68]; horizontal/vertical run-length histograms [103, 203] acquired from binary scale; entropy [216], slant/skew [55, 216, 233], co-occurrence matrix [98] attained from gray scale images, etc.

Srihari et al. [216] worked with the handwriting of 1500 U.S. individuals of various ethnic groups and age. From this database, they extracted several spatial features, subdivided into macro (e.g., entropy, slope, slant, etc.) and micro (e.g., gradient, structural, concavity, etc.) features [211].

In [146], Schomaker and Bulacu worked with PDFs (Probability-Density Function) or histograms of connected component contour (CO^3), edge direction, edge hinge/angle, where their hinge PDF outperformed many previous works. The same group came up with a modified version [149] and worked with PDFs obtained from contour-direction, contour-hinge, direction co-occurrence, run-length, grapheme codebook; and auto-correlation. Here also, the contour-hinge feature performed the best. Paraskevas et al. [68] achieved better performance from hinge feature while using the stroke skeleton instead of employing edge/contour as of [146, 149]. This hinge feature is quite effective for writer identification and has been generalized by He and Schomaker [202].

Jain and Doermann [181] worked with contour gradient descriptor from the segmented character which captured the local shape and curvature of the writing stroke. Their method won the "ICDAR 2013 competition on writer identification" [92].

Some local descriptors such as LBP (Local Binary Patterns) [64] and its variants [26, 27], SIFT (Scale-Invariant Feature Transform) [253], SURF (Speeded Up Robust Features) [153, 182], etc. are also used for writer identification. Newell and Griffin [18] proposed oBIF (oriented Basic Image Features) for writer identification and won two

Arabic writer identification competitions in conjunction with the ICDAR 2011 [16] and ICFHR 2012 [15].

Sometimes, the features are extracted by analyzing the ink-blob shapes. These shape features are mostly based on template [171] allograph [149], grapheme [3], fragment [205] of writing strokes. Sutanto et al. [171] used only one character 'a' to identify the writer. Writer identification was also performed on grapheme 'th' [243]. On characters 'd', 'y', 'f' and grapheme 'th', a work was reported in [244]. In [149], some allographic features were used, where at first, a codebook of graphemes was generated, followed by clustering, then the writer-specific grapheme scribbled PDF was calculated. In [3], the template matching of graphemes was used for writer identification. He et al. [205] generated codebook representation of stroke fragments at the junction to identify the writer. Siddiqi and Vincent [106] worked with morphologically similar writing stroke fragments.

The CNN (Convolutional Neural Network)-based features are currently very popular and also has been effectively used in writer identification [147, 198, 235, 256]. Here, the image pixel intensity is mapped to an embedded space through convolutions. Christlein et al. [235] proposed CNN induced features for writer identification. Tang et al. [256] applied the joint Bayesian model on CNN features to identify the writer. A deep CNN-based "DeepWriter" model was proposed by Xing et al. [147]. The CNN-based writer identification model of [198] outperformed the top three methods of "ICDAR 2011 Writer Identification Contest" [93].

The *transform* domain features are primarily made of frequency [1] and wavelet [260]. The examples of such transformation, used for writer identification-related feature extraction, are Fourier transform [1], DCT (Discrete Cosine Transform) [78], Radon transform [193], Gabor transform [85, 98], Hermite transform [17], DWT (Discrete Wavelet Transform) [199], etc. Sometimes, the local textural pattern is obtained in a transform domain, e.g., wavelet-domain LBP [138], Fourier phase spectrum-based LPQ (Local Phase Quantization) [64], etc.

In [260], a wavelet-based generalized Gaussian model was proposed, which was claimed to work better than traditional 2D Gabor filters for writer identification. The LPQ of [64] also worked better than Gabor transform.

Schomaker and Bulacu [146] showed lower performances of some wavelet-based descriptors, e.g., Haar, Odegard, Adelson, Antonini, Brislawn, Daubechies, Villasenor,

compared to their hinge feature.

Classifier : For writer identification, the distance-based similarity metric [59, 236], nearest neighbor classifier [26, 66, 68, 146, 149, 211, 216], kNN (k-Nearest Neighbors) [33, 52, 55, 181, 233] classifier are widely used for measuring the distance of query sample from the known samples. Here, various distance metric such as Manhattan [59, 68], Euclidean [68, 159], chi-square [68, 149], Mahalanobis [159], Hamming [146], etc. are popularly used. Sometimes, weighted distance function is employed, e.g., weighted Euclidean [246, 257], weighted chi-square [246], etc. In [146], with the nearest neighbor classifier, the chi-square worked best, and the hamming distance worked comparably well, however, Euclidean distance did not perform so well. Some other distance measures, such as cosine distance [5, 197, 236], Kullback-Leibler distance [261] are also found in the literature.

Some statistical models, e.g., Bayesian [106, 168], GMM (Gaussian Mixture Model) [237], HMM (Hidden Markov Model) [32], etc. are used for writer identification. The group of H. Bunke[5] used HMM for writer identification [28, 29, 32]. They also employed GMM for this task in [30].

The SVM (Support Vector Machine) [17, 64], neural network [233] have also been used in the literature. Nowadays, the rear part of the deep neural net [147, 198], which is basically an MLP (Multi-Layer Perceptron) [235], is also applied for writer identification.

1.4.1.2 Writer Verification

The past research on writer verification mostly employed the features used for writer identification. Thus, the writer verification task also uses spatial [8, 64, 101, 108, 137, 150, 155, 184, 215, 250] and transform [64, 84] domain features similar to the writer identification. Actually, the basic difference between verification and identification rests in the classification strategy. The writer identification task can be perceived as a multi-class classification problem, whereas the writer verification can be treated as a binary classification [155].

For writer verification, the classical Neyman-Pearson framework [122] is used with several distance-based similarity measures, such as Euclidean [8, 247], Mahalanobis [108], Hamming [149, 150], chi-square [84, 155], Earth mover's distance [250], etc. Correlation similarity [101] has also been used for this task in the past.

[5]Research Group on Computer Vision and Artificial Intelligence, University of Bern.

Among statistical models, the Bayesian model [6, 94, 215], HMM [28, 32], GMM [31], etc. have been employed. For writer verification, Schlapbach and Bunke [31] compared the use of HMM and GMM, where the HMM-based system worked better than GMM on a skillfully forged test set.

The neural network [216] and SVM [64, 137, 184] have also been reported to be used for writer verification. Nowadays, deep learning-based similarity metrics (e.g., Siamese net [111], triplet network [73]) are being used for several biometric verification tasks, but yet to be reported in the literature for writer verification task.

1.4.1.3 Writer Demographics

Writer demographic analysis is required to place a writer in one of the broader groups. Such analysis is sometimes essential to shorten the search space of the suspects. The past studies on demographics of writers were performed in several aspects such as gender [38, 53, 54, 105, 126, 126, 135, 161, 189, 251], handedness (left or right) [53, 135, 161], age-range [135, 161, 189], cultural background [189, 217], etc.

In [105], for gender classification, spatial features, e.g., slant, curvature, fractal, LBP-based features were used with neural net and SVM. Akbari et al. [251] used a transform domain feature, i.e., wavelet, with neural net and SVM for gender detection. Al-Maadeed et al. [189] used direction, curvature, tortuosity, contour/edge-based feature with random forest and kernel discriminant analysis for gender, handedness, age, and nationality analysis. In [161], the HOG and gradient LBP features-based SVM model was used for age, gender and handedness prediction. Bandi and Srihari [135] used document macro features with a single feed-forward neural net, and boosting/bagging-based neural net combinations for gender, handedness, and age classification. Nag et al. [217] used a textural-based curvature free feature with SVM to find the ethnicity/nationality of the writers among five nationalities of Bangladesh, China, India, Malaysia, and Iran.

In conjunction with ICFHR-2016, an international competition [53] was held on writer demographics analysis, where two subtasks on gender and handedness were focused. For gender classification, the winning method “MCS NUST” used some texture features such as LBP, HOG (Histogram of Oriented Gradients), gray-level co-occurrence matrix, etc. with SVM. For handedness classification, the “Nuremberg” method won the task by using VLAD encoded Contour-Zernike Moments with a combination of SVM and Random forest.

In another competition [54] at ICDAR-2015, a subtask was given on gender classification, where the winning method “CVC” used an LBP variant with PCA (Principal

Component Analysis).

1.4.1.4 Some Interesting Work on Writer Inspection

The above-mentioned works are usual state-of-the-art techniques for writer inspection from the conventional handwritten page. However, writer inspection is possible from some unconventional scribblings, such as musical score [12, 13], doodle, sketch [100], struck-out stroke, etc. The writer inspection research from the musical score is quite established [11], but others are yet to be explored.

Some research works on writer inspection have shown interesting results. Brink et al. [9] showed that the average writing slant is "much less important" for writer verification, where direction, hinge and grapheme-based features were used.

Chen and Lopresti [113] unveiled that retaining the page skewness achieved better writer identification performance than employing any deskew method. In [112], Chen and Lopresti also showed that preserving the ruling lines instead of removing those, improved the writer identification performance. In [113] and [112], the contour-hinge feature with SVM was used for writer identification. In [114], Chen et al. elaborated these experiments on "conservative preprocessing of document images" for writer inspection.

1.4.2 Writing Stroke Understanding

In the literature on document image analysis, very few papers are available for writing stroke understanding.

Chou and Yu [210] considered the quality of offline Chinese character stroke for sorting the characters of a database. They used character skeleton-based features with Bayes' decision rule for this task. It was an early attempt for character sorting in terms of character formation quality.

Majumdar et al. [23] worked with some coarse and fine level features for visual aesthetic analysis of the handwritten document. They employed a linear SVM to classify the documents into five aesthetic classes.

Bouletreau et al. [234] employed fractal behavior of writing strokes to generate some synthetic parameters with the objective of handwriting classification.

1.4.3 Content Understanding

We did not come across any work on content understanding performed directly from the handwritten document images.

1.4.4 Handwriting Database

In the field of document image processing, several offline handwriting databases exist. In this section, we mention some popular databases. These databases can be broadly categorized with respect to scripts, such as Alphabetic [4, 60, 72, 81, 91–93, 149, 216, 232], Abjad [15, 16, 24, 86, 156, 181, 191], Logographic [58, 104, 125, 223], and Abugida [130, 172, 188, 219]. However, some databases contain handwritten pages of multiple scripts [2, 52, 166, 190, 205], e.g., QUWI [190] database contains handwritings of English and Arabic. Sometimes, handwritings of some databases have common entries, i.e., the same handwritten sample exists in multiple databases, e.g., PBOK-Bengali [2], ICDAR-2013 segmentation-Bengali [166], and RoyDB-Bengali [172].

Table 1.1: Some popular offline handwriting databases

Database name	**Language/Script**	**# Writers**	**# Samples**
IAM (ver. 3) [232]	English	657	1539
CEDAR [216]	English	1500	4500
Firemaker [149]	Dutch	250	1000
RIMES [72]	French	1300	12723
PSI [4]	French	88	88
BFL [60]	Brazilian Portuguese	315	945
CVL [81]	English, German	310	1604
ICDAR-2011 EGFG [93]	English, Greek, French, German	26	208
ICFHR-2012 EG [91]	English, Greek	100	400
ICDAR-2013 EG [92]	English, Greek	250	1000
AHTID-MW [24, 86]	Arabic	53	22896 (#word)
KHATT [191]	Arabic	1000	6000
MADCAT-Arabic [181]	Arabic	316	3160
IFN/ENIT [156]	Arabic	411	26459 (#word)
ICDAR-2011 Arabic [16]	Arabic	54	162
ICFHR-2012 Arabic [15]	Arabic	206	618
HIT-MW [223]	Chinese	780	853
SYSU [125]	Chinese	245	950 (#character)
CASIA-Offline [58]	Chinese	1019	5091
KAIST-Hanja1 [104]	Hanja	200	156600 (#character)
CMATERdb1.1.1 [188]	Bengali	-	100
RoyDB-Bengali [172]	Bengali	60	17091 (#word)
RoyDB-Devanagari [172]	Devanagari	-	16128 (#word)
IIIT-HW-Dev [130]	Devanagari	12	95381 (#word)
TamilDB [219]	Tamil	-	26500 (#word)
ICDAR-2013 segmentation [166]	English, Greek, Bengali	-	350
PBOK [2]	Persian, Bengali, Oriya, Kannada	436	707
QUWI [190]	English, Arabic	1017	4068
EUHD [52]	English, Urdu	44	176
MSHD [52]	French, Arabic	87	1044
CERUG [205]	Chinese, English	105	420

In Table 1.1, we present the facets of some popular handwriting databases.

1.5 Organization of the Thesis

This thesis is broadly categorized into three parts with respect to the research contributions. The *first* part is about understanding the writer, whereas, the *second* part comprises of understanding the implicit characteristics of writing strokes. Finally, the *third* part deals with understanding the content revealing entities from handwriting.

The thesis is organized into ten chapters. The synopsizes of the chapters are mentioned as follow.

Chapter 1: This chapter contains an introduction and a summary of the major contributions of this thesis. A survey of relevant past research is also presented here.

Part I: Understanding the Writer : This part is dealt with *Chapter*s 2-5, as follows.

- ***Chapter 2***: A new method for writer verification is proposed in this chapter.
- ***Chapter 3***: This chapter describes writer identification, where training is performed on one script and testing is executed on another.
- ***Chapter 4***: The writer identification and verification on intra-variable handwriting are tackled in this chapter.
- ***Chapter 5***: This chapter discusses the writer identification performance while a handwritten document contains struck-out texts.

Part II: Understanding Writing Strokes : This part consists of *Chapter*s 6-8.

- ***Chapter 6***: The legibility and aesthetic analysis of handwriting are explored in this chapter.
- ***Chapter 7***: This chapter considers analyzing writing/reading difficulties of some textual characters.
- ***Chapter 8***: This chapter studies handwriting idiosyncrasy and its use in some problems.

Part III: Understanding the Content : This part contains only *Chapter* 9.

- ***Chapter 9***: This chapter approaches the content understanding of a handwritten document.

Chapter 10: This chapter concludes this thesis with a mention of the future scope of this research.

Part I

Understanding the Writer

CHAPTER 2

WRITER VERIFICATION BY *PDF-CNN* & SIAMESE NET

Automated writer authentication has been established within a legalized civic society [186], and commonly accepted in forensics to authenticate a person. This field of research is still challenging due to ever-increasing complex patterns. As a matter of fact, the handwritten samples of a person vary extensively with respect to the circumstances [47]. This makes the task even more challenging.

A detailed survey related to handwriting authentication up to the year 1989 is reported in [186]. Recent advancements of writer identification and verification are discussed in [144, 149]. In this chapter, we deal with the investigation of offline handwriting. A comprehensive survey dedicated to offline techniques is reported by Xiong et al. in [252]. We also discussed the literature review on writer verification in *Section* 1.4 of *Chapter* 1.

In this chapter, for writer verification, we focus on the handwriting of a complex Indic script, *Bengali* (endonym, *Bangla*) [178]. The Bengali script contains 50 basic characters and more than 250 compound characters [37]. A compound character is the conjunction of two or more (up to four) basic characters. A basic character can be conjoined with another in four possible major neighborhood directions (top, bottom, left, and right), according to Bengali orthographic rules. After the conjunction, the shapes of the characters may

This work is partially published as:
C. Adak, S. Marinai, B. B. Chaudhuri, M. Blumenstein, "Offline Bengali Writer Verification by PDF-CNN and Siamese Net", Proc. 13th IAPR Int. Workshop on Document Analysis Systems (DAS), pp. 381-386, Vienna, Austria, 24-27 Apr., 2018.

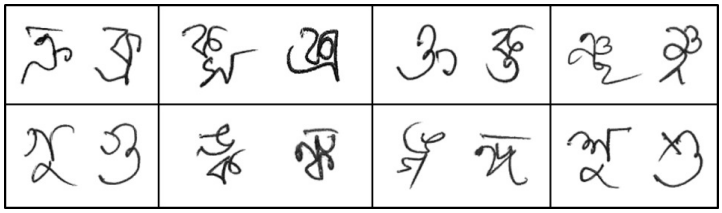


Figure 2.1: Every pair in a box comprises the same compound Bengali character written in two different styles [44].

change completely, or partially. Also, a particular character may be written in multiple ways (refer to Figure 2.1). Moreover, the individual characters of a word are mostly joined by a headline-like stroke called "*matra*". Besides the above, the cursiveness of Bengali writing makes it even more complicated [45]. Some more complex characteristics of the Bengali script can be found in [37, 44].

A brief discussion on offline Indic as well as Bengali writer identification/verification is reported in [44]. The earliest attempt on Bengali writer recognition was performed by Tripathi et al. [174] using some simple word bounding-box level micro-features. In [226], a 2-D autoregressive (AR) model was used for Bengali and French writer identification. A directional and gradient feature-based SVM model was used by Chanda et al. [194] to identify Bengali writers. In [192], Biswas and Das used the occurrence of text component structure and a rejection schema to investigate Bengali writers. A Radon transform projection-based improved version of [192] was reported in [193]. In [44], some textural features were used to identify writers from isolated Bengali characters and numerals.

In one common approach, the writer verification task is addressed as a "one-to-one comparison" to decide whether two samples are written by the same writer [149]. In other words, the writer verification can be perceived as a binary classification problem to answer *yes*/*no* to the question whether Doc-A is written by Writer-B [47].

In this chapter, for writer verification, we perform this classification on the basis of the PDF (Probability Distribution Function) of some handcrafted features. We intend to hybridize handcrafted features with auto-derived features to investigate their effect on verification performance. To this purpose, we feed the PDF into a CNN (Convolutional Neural Network) to produce auto-derived CNN features from the PDF information. Then, the CNN features are fed into two different classification modules: one MLP (Multi-Layer Perceptron) and one Siamese neural network for writer verification.

In *Section* 2.1, we describe our proposed method in detail. Then, the experimental results are presented in *Section* 2.2. Finally, *Section* 2.3 summarizes this chapter.

2.1 Proposed Method

Once a digital image of a handwritten page is obtained, it undergoes a number of preprocessing steps. The scanned digital image of a handwritten page sample may be speckled with some noise. In the preprocessing stage, very small components are removed by a moderately fast single pass connected component labeling method [87]. If the document contains any doodle/drawing-like non-textual component, this non-textual component is removed using the technique presented in [42]. Some struck-out/crossed-out texts may exist in the document that also need to be removed due to their impeding effect on the writer analysis task [48]. We use the method in [36] to remove struck-out/crossed-out texts from the document. Subsequently, the preprocessed digital image of a handwritten page is finally used for writer verification.

As mentioned earlier in this chapter, the writer verification task can be expressed as a binary classification problem. To deal with the classification problem of writer verification, at first, we generate some handcrafted [142] features from the handwritten pattern. In the next subsections, we discuss the details of the features used for this work and Subsequently describe the classification methods.

2.1.1 Handcrafted Feature PDF

Instead of employing the traditional single-valued feature to capture the facet of writing individuality, here we employ a probability distribution function (PDF) as an entire feature vector [149]. Two different types of handcrafted features namely *textural* and *allographic* are used and described in the following.

2.1.1.1 Textural Feature PDF

The textural feature usually provides information of pen-grip holding style and writing slant. It has been found experimentally that among the textural features such as directional, run-length, hinge, gray-level co-occurrence, the contour-hinge feature works well for writer verification [149]. We, therefore, focus on the contour-hinge feature (f_{ch}) here, as described below.

A hinge is created when two contour fragments are joined to a common junction-pixel [149]. Consider two contour fragments making angles ϕ_1 and ϕ_2 respectively with the horizontal axis. To avoid redundancy, we assume $\phi_2 \geq \phi_1$. The angles are spanned to all four quadrants (360^o) centering the junction-pixel. The entire quadrant is divided into $2n$ parts, where n is a given parameter. Based on this division, a normalized histogram

is generated with a joint probability distribution $p_f(\phi_1, \phi_2)$, as demonstrated in [149]. Although the total number of combinations of two angles is $4n^2 = 2n \times 2n$, the final number of combinations is reduced to ${}^{2n}C_2 + 2n = n(2n+1)$, due to the assumption of non-redundancy. Here, ${}^{2n}C_2$ refers to the number of ways we can choose 2 elements out of $2n$ elements and the greater value is always assigned to ϕ_2 due to our assumption; $2n$ refers to the case when ϕ_1 and ϕ_2 lie in the same partition of the quadrant. In our experiment, empirically we choose $n = 12$, leading to $12 \times (2 \times 12 + 1) = 300$ combinations. Though we have used the matrix of size 24×24 in our experiment, we consider only the non-redundant entries ($\phi_2 \geq \phi_1$). In other words, the employed feature map is in the form of a triangular matrix [149].

2.1.1.2 Allographic Feature PDF

The allographic feature divulges the character shape and formation [149]. It is assumed that a writer acts as a stochastic generator of grapheme/fragment (small parts of character) shapes. The probability distribution of such grapheme shapes may be used as a feature. In this work, the handwritten text is segmented into graphemes/fragments by means of the water reservoir technique [227] and by partitioning the ink-trace at the minima of the lower contour vertically thorough the ink-stroke-width of the upper contour [149].

The grapheme codebook generation is obtained by using Kohonen's 2-Dimensional SOM (Self-Organizing Map) [220]. Here, we use a 25×25 SOM-2D map. Some alternatives for codebook generation are k-means and one-dimensional SOM. However, we choose the SOM-2D because of the spatial organization of codebooks that has been used in some document image analysis tasks [213, 214] and that can be taken into account by local receptive fields in convolutional layers of the CNN architecture.

The feature vector, in this case, is a probability distribution function of graphemes organized into the 2D topology of the SOM. A histogram of 625 ($= 25 \times 25$) bins is therefore generated considering every grapheme in one bin. A sample grapheme is matched with the nearest codebook prototype and is put in a histogram bin, similarly to [149]. From this histogram, we obtain a normalized feature distribution map of size 25×25.

Figure 2.2: PDF-CNN model diagram.

2.1.2 Handcrafted Feature PDF Hybridized with Auto-derived CNN Features (PDF-CNN)

In a CNN architecture, generally, an image is used as an input to extract some auto-derived features. Here, the handcrafted feature PDF is fed into the CNN, since it is still one 2D matrix of non-negative values. We call this CNN architecture as a *PDF-CNN*. Figure 2.2 shows the model diagram of a PDF-CNN.

The PDF-CNNs used for writer verification are discussed below.

2.1.2.1 Textural PDF-CNN

The textural contour-hinge PDF, as demonstrated in *Section* 2.1.1.1, is the input to the CNN for deriving an automated feature vector. As discussed earlier, our square textural hinge map is of size 24×24, and half of this matrix (below its main diagonal) is left unused to avoid redundancy.

The CNN architecture contains two convolutional layers, each followed by a sub-sampling layer. The overall architecture is shown in Figure 2.3:(a).

The convolutional layer C_1 has 16 feature maps of size 24×24. Each feature map is connected to a 3×3 (N_F) neighborhood in the input textural PDF map. The MP_1 subsampling (max-pooling) layer contains 16 feature maps of size 12×12, each connected to a 2×2 (N_K) neighborhood in the corresponding feature map in C_1.

The following C_2 convolutional layer consists of 32 feature maps of size 12×12. Here, $N_F = 3 \times 3$, therefore, each feature map is connected to a 3×3 neighborhood of corresponding MP_1's feature map. The MP_2 sub-sampling (max-pooling) layer contains 32 feature maps of size 6×6. Here, $N_K = 2 \times 2$.

The following layer FC has 1152 feature maps each sized 1×1 and fully connected with MP_2.

The ReLU (Rectified Linear Unit) is used as the activation function for all the convolutional layers. A learning rate of 10^{-3} with a momentum term of 0.9 is used for training up to 500 epochs. We have obtained an 1152-dimensional feature vector from this textural PDF-CNN.

2.1.2.2 Allographic PDF-CNN

The handcrafted allographic PDF feature map of size 25×25 (refer to *Section* 2.1.1.2) is fed to the CNN for automated feature extraction.

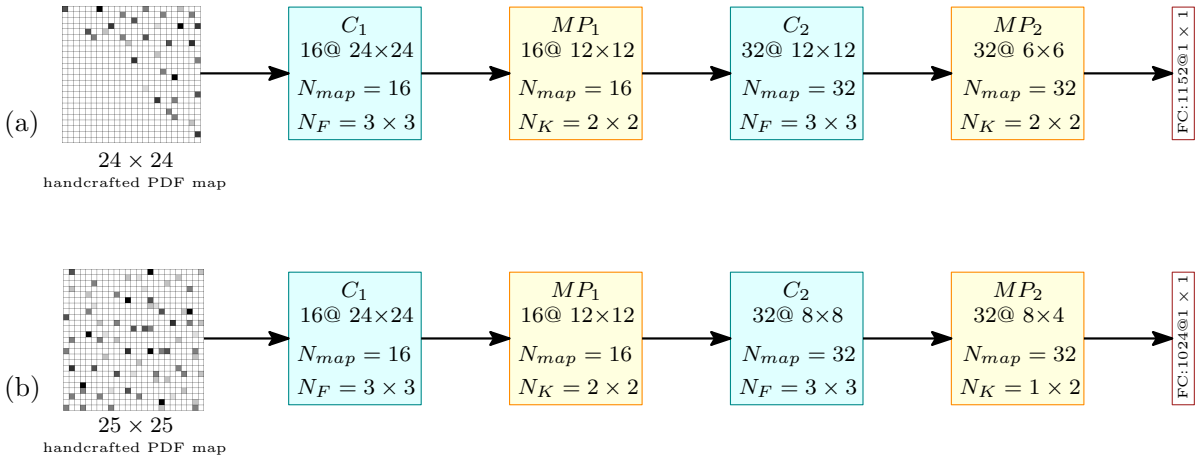


Figure 2.3: (a) Textural PDF-CNN and (b) Allographic PDF-CNN architectures.

This CNN architecture is also constructed with two convolutional layers, each followed by a sub-sampling layer, as shown in Figure 2.3:(b). The first two layers (C_1 and MP_1) of CNN architecture are analogous to the Textural PDF-CNN (refer to *Section* 2.1.2.1).

The following C_2 convolutional layer consists of 32 feature maps of size 8×8, and again $N_F = 3 \times 3$. The MP_2 sub-sampling (max-pooling) layer contains 32 feature maps of size 8×4 and now $N_K = 1 \times 2$ is used.

In this case, the following layer FC has 1024 feature maps, each of size 1×1 and fully connected with MP_2.

Also, for this CNN, the ReLU is used as an activation function for the convolutional layers. The learning rate is set to 10^{-3} with a momentum of 0.9. The training epochs are increased up to 500. This allographic PDF-CNN produces a 1024-dimensional feature vector.

2.1.3 Classification / Writer Verification

The output of a PDF-CNN is fed into a classifier for writer verification. It is perceived as a binary classification task to classify/authenticate a writer in the “same” (*yes*) or “not same” (*no*) class. We choose MLP (Multi-Layer Perceptron) and Siamese neural network separately for this verification task.

2.1.3.1 CNN-MLP

The output of the PDF-CNN of *Section* 2.1.2 is fed into an MLP classifier which contains one hidden layer and two neurons as MLP output. One different network is trained

for each writer and the target output has two different outcomes for samples of the corresponding writer and for remaining texts.

The training is made with the scaled conjugate gradient backpropagation, since it requires less memory and the mean squared error is used as the performance parameter. The number of neurons in the hidden layer has been empirically set from 512 to 100 on the basis of preliminary experiments. The activation function for the MLP is the hyperbolic-tangent sigmoid. The training epochs are increased from 1,000 up to 10,000.

2.1.3.1.1 Textural CNN-MLP : As discussed in *Section* 2.1.2.1, the handcrafted textural feature PDF is fed to the CNN. From the textural PDF-CNN, we have obtained an 1152-dimensional feature vector. This feature vector is fed to the above-mentioned MLP of *Section* 2.1.3.1 for writer verification.

2.1.3.1.2 Allographic CNN-MLP : Similarly, the handcrafted allographic feature PDF is sent as input to the CNN to obtain a 1024-dimensional feature vector (refer to *Section* 2.1.2.2). This feature vector is input to the above-mentioned MLP of *Section* 2.1.3.1 for writer verification.

2.1.3.2 CNN-Siamese

Siamese neural network is a well-known architecture, used for ranking the similarity between two inputs [90, 111] and for weakly supervised learning. Here, we use the Siamese net for writer verification to know whether a handwritten sample is written by a particular writer. The PDF-CNN, generated in *Section* 2.1.2, is employed in the Siamese neural network.

The usual Siamese neural network contains two identical subnetworks. Our PDF-CNN depicts both subnetworks of the Siamese twins, individually. The Siamese twin subnets are joined with a loss function to calculate the similarity between the feature representations on each subnet. The contrastive loss function [195] is used here for training the whole network according to the similarity of the two handwritten samples.

2.1.3.2.1 Textural CNN-Siamese : The handcrafted textural feature PDF is fed into the CNN and the textural PDF-CNN is the architecture of both Siamese twins. The contrastive loss function [195] is used here as mentioned earlier in *Section* 2.1.3.2.

2.1.3.2.2 Allographic CNN-Siamese : The handcrafted allographic feature PDF is fed into the CNN (refer to *Section* 2.1.2.2). This allographic PDF-CNN is one subnet of the Siamese twins, whereas the other one is exactly identical. Here, also, the contrastive loss function [195] is employed.

2.2 Experiments and Discussion

Before evaluating the performance of our system, we discuss the database used in our experiments.

2.2.1 Database Employed

We employed a dataset of 100 writers being in the age group of 9 to 67 years old with different academic backgrounds from primary school to university level. Each writer produced 3 pages of Bengali handwriting. Therefore, a total of 300 document pages were stored in this database. The writers freely wrote a piece of Bengali text independent of any content. The dataset was made in an uncontrolled way. The writers were free to write anytime without any time-gap restriction between two writing samples. The A4 sized 75 GSM (g/m^2) blank white pages without any ruled-lines and a 0.5 mm ball-point black inked pen of the same model were provided to the writers for maintaining consistent writing strokes.

We split each page roughly into two parts, which is an approach commonly used in the literature [248] for dividing the samples among training, validation and test set. Since every person wrote 3 pages, 6 handwritten sub-samples per person were obtained.

2.2.2 Results and Evaluation

In this subsection, we present the experimental results and analyze the performance of our methodology.

We had 6 handwritten sub-samples for every writer. The training and validation sets contained 3 and 1 samples of each writer, respectively. The remaining portion of the dataset was used for testing. In other words, the training, validation and test sets were in the ratio of 3:1:2. All parameters of our system were tuned on the validation set.

The system performance was evaluated in terms of *False Acceptance Rate* (FAR) and *False Rejection Rate* (FRR) [122]. The *Balanced Error Rate* (BER) is the arithmetic mean

of FAR and FRR. In other words, BER = (FAR+FRR)/2. The *Balanced Accuracy* can be calculated as (100−BER)%.

The *Equal Error Rate* (EER) can also be obtained where the FAR is equal to FRR in the plot of error rate vs. decision threshold (T). In the case of the CNN-MLP, the threshold T is used to decide whether one unknown handwritten fragment belongs to the class for which the CNN-MLP network has been trained. In the case of the CNN-Siamese, the T value is used to decide whether two handwritten fragments belong to the same writer. We present these error rate curves with respect to different values of T in Figure 2.4 - Figure 2.7.

The system performances with respect to EER and balanced accuracy are shown in

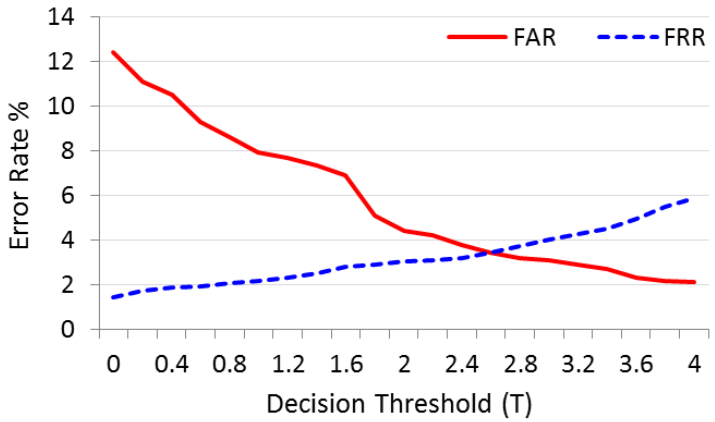


Figure 2.4: Textural CNN-MLP performance evaluation: FAR and FRR plot. EER = 3.42.

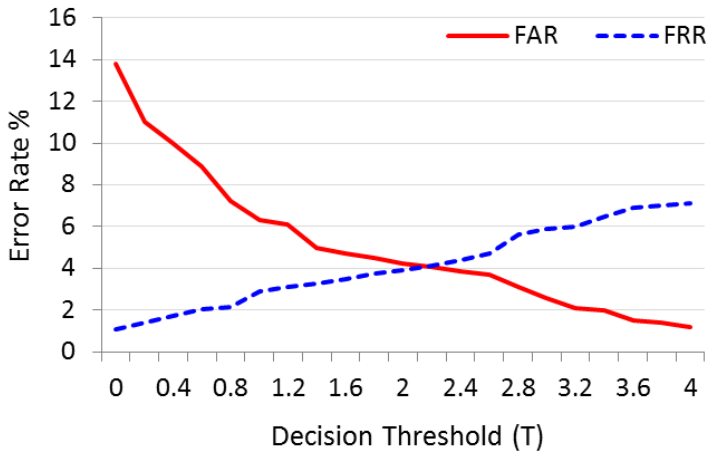


Figure 2.5: Allographic CNN-MLP performance evaluation: FAR and FRR plot. EER = 4.07.

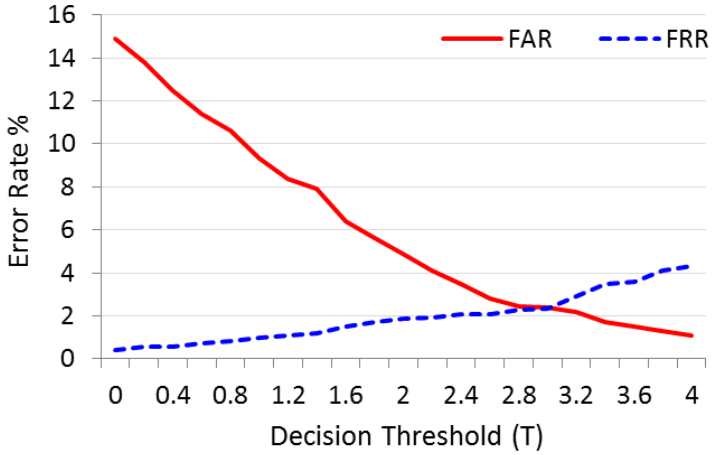


Figure 2.6: Textural CNN-Siamese performance evaluation: FAR and FRR plot. EER = 2.36.

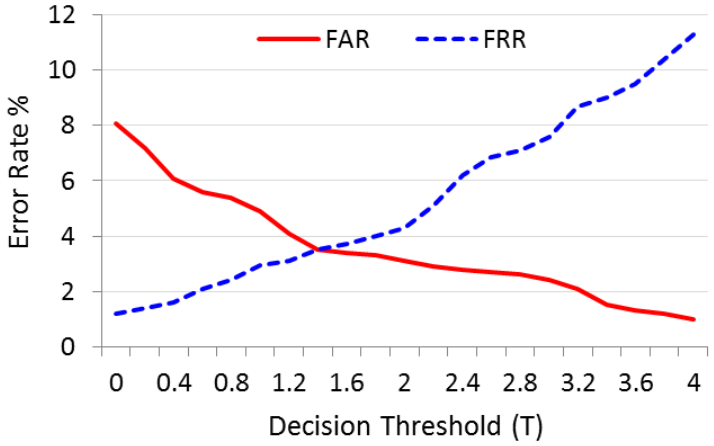


Figure 2.7: Allographic CNN-Siamese performance evaluation: FAR and FRR plot. EER = 3.51.

Table 2.1.

Testing on our Bengali offline dataset, overall the CNN-Siamese performed better than the CNN-MLP. The textural feature worked better than the allographic ones. In particular, the textural PDF-CNN with Siamese net provided best EER of 2.36% on our test set. The lowest performance, obtained from allographic PDF-CNN with MLP, was even well, i.e. 4.07% of EER (refer to Table 2.2).

It was interesting to notice that we did not use very deep neural architecture because of the small-sized PDF feature maps. However, our system performed satisfactorily with respect to the computation and space cost efficiency.

Table 2.1: Writer verification performance

Method	***Equal Error Rate (%)***	***Balanced Accuracy (%)***
Textural CNN-MLP	3.42	96.49
Allographic CNN-MLP	4.07	95.92
Textural CNN-Siamese	2.36	97.64
Allographic CNN-Siamese	3.51	96.47

2.2.3 Comparison

To the best of our knowledge, the number of research papers on offline Bengali writer *identification* [44, 174, 192–194, 226] is quite limited and we have not found any comparable work on Bengali writer verification.

The benchmark writer verification system like [149] produced 4.1% of the best EER on Firemaker-uppercase Dutch dataset using the textural contour-hinge feature. Bulacu et al. [149] also obtained 2.6% best EER on the Large-lowercase (English+Dutch) database by combining textural (contour-hinge, run-length) and allographic features. In [149], the system performance was reported on the Dutch and English script. In this chapter, our emphasis is on Bengali which is a more complex script. Here, we have obtained 2.36% of the best EER using a textural PDF-CNN with a Siamese net.

In [184], the overall error rate was 3.9% on the BFL (Brazilian Forensic Letter) database by employing texture-based features. Bensefia et al. [5] obtained 95.66% balanced accuracy on the English IAM database. In our current Bengali writer verification task, the lowest balanced accuracy is 95.92%, obtained using an allographic PDF-CNN with an MLP.

Pal et al. [218] obtained 33.82% of EER using uniform local binary pattern (LBP) on the Bengali part of the *BHSig260* signature verification dataset. Although we are not focusing on signatures, this *BHSig260* database can be treated as a word database containing mostly full signatures. In our case, we worked at the paragraph level writer verification and obtained the best result of 2.36% EER on our Bengali database.

To perform some additional comparisons, we executed some of these state-of-the-art methods [149, 184, 218] on our Bengali database (refer to *Section* 2.2.1). Their performances in terms of EER and balanced accuracy are shown in Table 2.2.

Here, our lowest performing method (Allographic CNN-MLP) even worked better than some earlier methods in the literature [149, 184, 218].

Table 2.2: Comparison with some state-of-the-art methods

Method	***Equal Error Rate (%)***	***Balanced Accuracy (%)***
Contour-hinge [149]	5.68	94.28
Texture [184]	6.39	93.60
LBP [218]	6.12	93.84
Ours inferior (Allographic CNN-MLP)	4.07	95.92

2.3 Summary

In this chapter, we propose a method for offline writer verification on a fairly complex script, Bengali. Here, we hybridize handcrafted feature (both textural and allographic)-PDF with auto-derived CNN features. These hybrid features are finally fed into an MLP and a Siamese-twin net separately. The textural contour-hinge PDF induced CNN with a Siamese net, i.e., textural CNN-Siamese, has performed the best by achieving an EER of 2.36% on our Bengali database. This database contains 300 Bengali pages written by 100 volunteers. The textural CNN-MLP, allographic CNN-MLP, and allographic CNN-Siamese have produced 3.42%, 4.07%, and 3.51% EER, respectively.

In the future, we plan to undertake further experiments on the different datasets. We also plan to hybridize different neural networks for a comparative study.

CHAPTER 3

WRITER IDENTIFICATION ACROSS SCRIPTS

Handwriting analysis is considered as a challenging task in pattern recognition due to its wide variability. Such writing variation occurs between persons over time, geographic locations and cultural aspects. Sometimes, the writing of an individual varies with aging, mental state, writing speed, writing tool, and so on, as discussed earlier in *Chapter* 1 . Therefore, writer identification and verification, which are sometimes required in computational forensics, are important and useful tasks of practical concern.

Writer verification can be viewed as a binary classification problem, where the classifier decides whether a manuscript is written by a particular writer or not. Writer identification is perceived as an n-class classification problem, where a handwritten specimen is provided and the task is to determine its writer from a set of n possible writers.

A survey on classical methods of writer inspection is discussed in *Section* 1.4 of *Chapter* 1. In [149], Balacu and Schomaker have divided the task of writer verification/identification into two general groups: text dependent and text independent. Text dependent [75] identification/verification is basically performed on known semantic content, whereas a text independent method is, as its name suggests, unfettered by text content. This text independent [149, 248] identification is essential because, at the time

This work is partially published as:
C. Adak, B. B. Chaudhuri, M. Blumenstein, "Writer Identification by Training on One Script but Testing on Another", Proc. 23rd Int. Conference on Pattern Recognition (ICPR), pp. 1148-1153, Cancun, Mexico, 4-8 Dec., 2016.

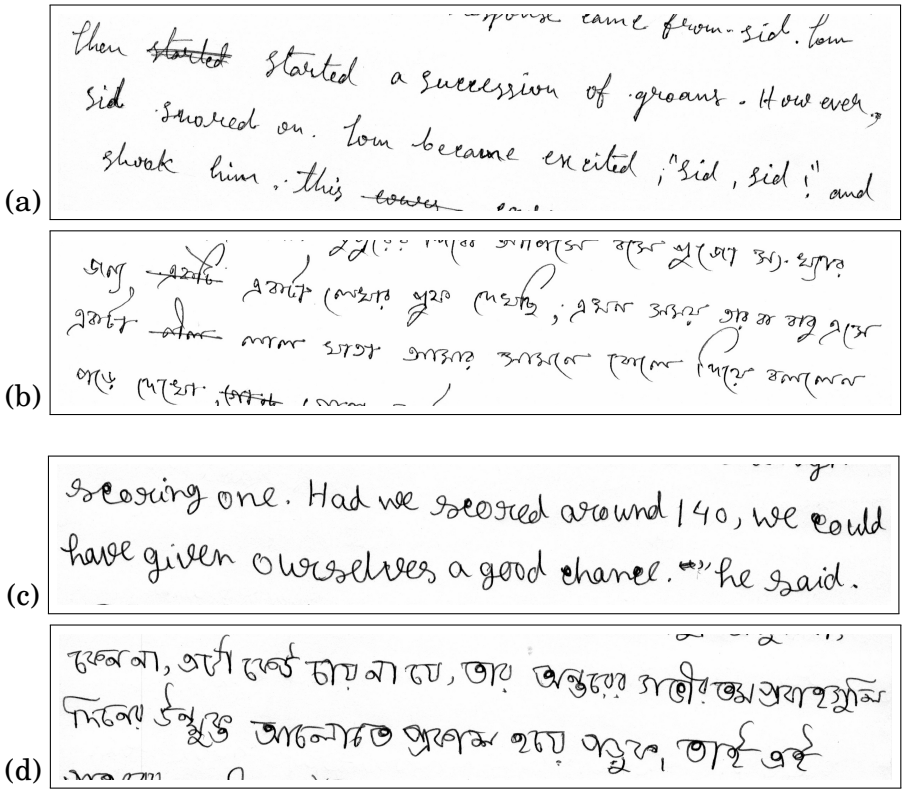


Figure 3.1: Handwriting specimens of two separate writers in both (a),(c) English and (b),(d) Bengali script.

of handwriting matching, the same textual content written by the same person may not be available. It may also be a requirement to analyze some unknown texts written by a certain individual. Here, we deal with text independent writer identification.

In a multilingual country, a writer may write in multiple (mostly two) scripts. Therefore, script independent writer identification is also essential. In ICDAR-2011, a contest [93] was organized on writer identification, where the participants were given handwriting of a writer in four different languages: English, French, German and Greek. Wu et al. [248] used this ICDAR-2011 dataset in their SIFT-based writer identification approach and obtained fairly good results. In that work [248], samples from all four languages were used for training.

Now, a situation may arise when we have the knowledge of a writer's handwriting only in one script (say, Sc_1) and we want to identify/verify the writer using test data in another script (say, Sc_2). In this chapter, writer identification/verification is performed across scripts. Here, we have taken this problem where the Bengali (Sc_1) handwriting of a writer is available for training and we want to identify/verify this writer by testing

on some English (Sc_2) handwriting, and vice-versa. This problem is interesting, since Bengali is an *Abugida/Alpha-syllabary* Indic script, while English is an *Alphabetic* Latin script. The structures of these scripts [37] are dissimilar due to their difference in origin. However, we believe that there are some implicit personal characteristics in handwriting styles, which are script independent. In Figure 3.1:(a),(b), the English and Bengali handwritten samples of an individual are shown. One more pair of samples scribbled by another writer is also shown in Figure 3.1:(c),(d).

In West Bengal (India) and Bangladesh, the written medium of education is mostly Bengali and English. This region was under the British colonial power for about 200 years and hence the second language in school/college education is English. The same situation may occur in many countries of Europe, where writing in English is popular besides French, German, Greek, Italian, Spanish, and so on. However, for such cases, the writer identification task is relatively easier since both English and native scripts are Roman-based only.

Ramaiah and Govindaraju [61] proposed an accent-based writer identification method of online handwriting which worked in two stages, namely "accent identification" followed by "writer identification". Although our work is focused on Bengali native writers, their work [61] is different from ours, because they have sufficient training data. However, in our approach training samples of a script (e.g., English) are considered unavailable. It uses some shared information from another script (e.g., Bengali) for training. We want to find out, given a training dataset (offline handwritten page of a writer) in Bengali script, whether it is possible to identify/verify the writer when we get his/her English writing for testing. Therefore, the main thrust of our study is to train our system with Bengali handwriting of an individual and to test it against the individual's English writing, and vice-versa. Here, we select some novel structural and statistical features based on the information from full text-pages, text-lines, words, characters, graphemes, and even pixels. These features are employed for the writer classification, i.e., identification and verification.

In this chapter, the contributions of our work are as follows.

(i) Tackling a need-based relatively new problem, that is, identifying a writer from an unknown script (e.g., English) using prior knowledge about another script (e.g., Bengali).

(ii) Presenting a study on some moderately novel latent features, both structural and statistical, which have an impact on writer identification while a different script is used for supervised training.

(iii) Analyzing the performance of various classifiers and their combinations related

to such a special type of writer identification/verification.

(iv) Generating a new handwritten database from 100 writers for an investigation into such problems with which this chapter is concerned.

The rest of the chapter proceeds as follows. *Section* 3.1 describes our proposed method in detail. Then *Section* 3.2 presents and analyzes the experimental results. Finally, *Section* 3.3 summarizes this chapter.

3.1 Proposed Method

Bengali is an alpha-syllabary script and English is an alphabetic script. There exist a good degree of differences among characters of these two scripts. Therefore, in this problem, we cannot employ character shape matching to identify the writer. However, a writer may imprint his latent identity in the margin/page-border, text-line skew, word slant, interline and inter-word gaps, etc., which may be nearly identical when the individual writes in either script. Also, sub-segments/fragments of characters in two different scripts may have some identical shapes. Some computational analysis based on writing direction and curvature may be useful for our purpose. We start with the extraction of various features of individual writing styles and then perform the classification with the objective of writer identification/verification. The details are as follows.

3.1.1 Feature Extraction

An elaborate description of the employed features in our problem is given below.

3.1.1.1 Margin Space Width

An individual, when writes on a four-sided blank white page, the starting position and the end position of each text-line is bounded by two-sided margins with white space. When the writing on a full page is completed, we can also find the upper and lower side margins. Now both Bengali and English scripts are written in a left-to-right horizontal direction. By inspecting a moderate number of handwriting, we note that more variation occurs at the left and top margins. Therefore, the left and top margins are more important compared to the right and bottom margins for writer identification.

In [212], several types of margins and their impact on the writer's mental condition are considered. But, in this work, we refrain from examining such graphological issues.

We only calculate the average white-border width outside the margin. The margins are detected using a simple projection profile based technique [165], which is effective for our purpose. The left/right margin space widths are calculated using a vertical histogram, whereas top/bottom margins are computed by a horizontal histogram. These margin space widths (left, top, right and bottom) are used as features f_1, f_2, f_3 and f_4.

The margin space width does not appear to be a robust feature, but inspired by [212], we employ margins as a latent feature for individual writing style. To investigate the impact of margins, we collect the text written on white A4 sized pages only (refer to *Section* 3.2.1).

3.1.1.2 Text-line Skewness

When written on a white blank page which does not contain any ruled line, the individual handwritten text-lines may not be horizontally aligned and may be skewed/oriented at some positive/negative angle.

Before skew detection, the document image should be segmented into text lines. For text-line segmentation, we use a state-of-the-art method [148], where a Gaussian window and a level set technique are used.

Now, a reasonably straight line $\mathscr{L}$ is considered horizontally through the middle of the segmented text-line. Next, the skew is measured as the angle between this line $\mathscr{L}$ and the horizontal axis. For a text-line, the skew may be upward or downward direction, that is, making a positive or negative slope with the horizontal axis. We take the average and standard deviation of skewed angles of all upward directed text-lines, and use as features f_5 and f_6, respectively. Similarly, for downward directed text-lines, we also obtain the average and standard deviation of the skew angles and employ these as features f_7 and f_8.

3.1.1.3 Word Slant

Word slant is like a shearing transformation over a straight word when it is transformed into up-right or up-left and rigidly attached to the baseline. Usually, an individual writes by maintaining his/her characteristic slant irrespective of scripts. Therefore, we use the slant-angle as a feature.

Before proceeding for word slant estimation, word segmentation is required. For this purpose, we use a 2D Gaussian filter on the document image. The standard deviation (σ) and block size ($height \times width$) for this filter is chosen automatically by analyzing the connected components in a text-line. Since the words are arranged horizontally in

Bengali and English scripts, the block size is chosen longer in width than its height. After blurring, an adaptive threshold is used for binarization. The connected components of this binarized image are considered as individual words.

The word slant can be calculated by the vertical parts of a word. In Bengali and English scripts, the characters having vertical parts help us to calculate the word slant. For this slant estimation, we use the method of [74]. This technique is based on the vertical projection profiles and the Wigner-Ville distribution. The average and standard deviation of the word-slants in a page are used as features f_9 and f_{10}, respectively.

3.1.1.4 Inter-textline and Inter-word Gap

Between two consecutive text-lines, a writer leaves some white space to make a demarcation, called the inter-textline gap. Similarly, between two consecutive words, an inter-word white gap is maintained. Such white gaps also vary with individual handwriting and reflect the characteristic of the writer.

From the text-line and word segmented image, we find the inter-textline and inter-word gaps. The average and standard deviation of inter-textline gaps of a page are calculated and used as features f_{11} and f_{12}, respectively. For words also, the inter-word gap average and standard deviation are obtained to use as features f_{13} and f_{14}, respectively.

3.1.1.5 Main-body Text Height

We notice that when a person writes either in Bengali or in English, the text main body height remains almost the same. For measuring the text main body height, we analyze individual words obtained from the segmented word image.

To detect the main body text height, we partition a word image into three zones (upper, middle and lower) by two imaginary lines called upper and lower baselines. These baselines are obtained by simple horizontal histogram projections. Some details of the approach can be found in [231]. The maximum number of object (ink) pixels generally lies in the middle zone, where the horizontal histogram profile reaches the global maximum. This middle zone is actually the main body of a word, whose height is calculated for our requirement. The average and standard deviation of main body heights of all words in a page are calculated and used as features f_{15} and f_{16}, respectively.

3.1.1.6 HOG at Keypoints

We intend to capture some common aspects of writing styles of Bengali and English scripts. Therefore, we intend to find some points of interest, i.e., keypoints p_i on the handwritten text image, since both the Bengali and English handwritten ink-strokes contain start/end-points, branch-points and curved-points [44]. Here, we obtain some structural and SIFT keypoints on the text-strokes similar to the scheme used in [44].

In the neighborhood of each keypoint p_i, a small window of size $n \times n$ is considered. In this window, we obtain a histogram of oriented gradients (HOG) [162] considering 8-bin angular ($360^o/45^o$) information. Here, the gradient strengths are normalized using $L2$-norm. We use this HOG descriptor as feature f_{17}, since it stores significant information about text-stroke gradient change. The dimension of the feature f_{17} is 8. We note a substantial change in the gradient that occurs for both Bengali and English handwritten strokes.

3.1.1.7 Direction and Curvature at Keypoints

On the ink-strokes, the above-mentioned keypoints of *Section* 3.1.1.6 are also used to find directional and curvature features. Here, we take the idea of feature extraction for "The NPen++ Recognizer" [209] and use it according to our requirement in the offline situation as follows.

Between two connected (with ink-stroke pixels) keypoints p_i and p_{i+1}, we calculate its *writing direction* in Cosine and Sine values, and employ them as features f_{18} and f_{19}, respectively.

$$f_{18} \equiv cos(\theta_i) = \frac{p_{i+1}.row - p_i.row}{d_i} \tag{3.1}$$

$$f_{19} \equiv sin(\theta_i) = \frac{p_{i+1}.col - p_i.col}{d_i} \tag{3.2}$$

where, $p_i.row$ and $p_i.col$ are the row and column indices of p_i, and $d_i = \sqrt{(p_{i+1}.row - p_i.row)^2 + (p_{i+1}.col - p_i.col)^2}$.

The *curvature* of the ink-stroke is basically the angle (Cosine and Sine) generated by lines from p_{i-1} to p_i and p_i to p_{i+1}, which are used as features f_{20} and f_{21}.

$$f_{20} \equiv cos(\theta_{i+1} - \theta_i) = cos\theta_{i+1}cos\theta_i + sin\theta_{i+1}sin\theta_i \tag{3.3}$$

$$f_{21} \equiv sin(\theta_{i+1} - \theta_i) = sin\theta_{i+1}cos\theta_i - cos\theta_{i+1}sin\theta_i \tag{3.4}$$

These features are acquired from each keypoint. We generate separate normalized histograms of f_{18}, f_{19}, f_{20}, f_{21} spanning in the range [-1, 1] for 20 number of bins. Therefore, the dimension of each of these feature vectors is 20.

3.1.1.8 Loop-like Shapes

Some Bengali and English characters contain loop-like structures. Also, some extra loop-like patterns may arise due to individual cursive writing styles. Therefore, the loop feature plays an important role in our problem. In [171], Sutanto et al. used loop-based features for discrimination of the handwritten character 'a'. Motivated by [171], we also adapt two types of loop feature.

In a loop region, let LP be the set of object pixels, and let LC be the set of pixels ($p \in LC$) on the boundary of this loop region. The counts of object pixels in LP and LC are denoted by $n(LP)$ and $n(LC)$, respectively. The center of gravity (CG) inside the loop region is ($CG.row$, $CG.col$) given by: $CG.row = \frac{1}{n(LP)} \sum_{p \in LP} p.row$; $CG.col = \frac{1}{n(LP)} \sum_{p \in LP} p.col$.

We measure the standard deviation (σ_L) of the loop as distinct from an ideal circle.

$$\sigma_L = \sqrt{\frac{1}{n(LC)} \sum_{p \in LC} (X_p - \mu_L)^2} \tag{3.5}$$

where, $X_p = \sqrt{(p.row - CG.row)^2 + (p.col - CG.col)^2}$ and $\mu_L = \frac{1}{n(LC)} \sum_{p \in LC} X_p$.

In a page, we obtain the average and standard deviation of all σ_L's, and use as feature f_{22} and f_{23}, respectively.

Finally, we calculate another loop-feature based on the *roundness* ($\mathscr{R}$) of an object.

$$\mathscr{R} = \frac{4\pi . n(LP)}{(\sum_{p \in LC} X_p)^2} \tag{3.6}$$

From a page, we obtain the average and standard deviation of $\mathscr{R}$'s, and use as feature f_{24} and f_{25}, respectively.

Thus, we have obtained these 25 types of features which are extracted from full page information, text-line/word/character level, and even semi-character/grapheme/point level information. Here, all the features are normalized. By counting the above-mentioned individual feature dimensions, we have a feature vector of size 108.

3.1.2 Classification

We formulate the writer verification as a *binary* classification problem and writer identification as an *n-class* classification task, as mentioned earlier in this chapter. For such

classification, we employ some well-known classifiers such as kNN (k-Nearest Neighbors), MLP (Multi-Layer Perceptron), and SVM (Support Vector Machine). Here, at the time of classifier training, one script (e.g., Bengali) is available; whereas for testing, another scripted (e.g., English) handwritten specimen is provided.

The input metrics of the classifiers are tuned and some pre-arrangements of classifiers are considered, such as the distance function of kNN, the training function and number of hidden layers/neurons for MLP, and also the kernel function and parameters of the SVM. We also use different sized subsets of training data in multiple sessions to design multiple classifiers. The classifier outputs are combined using a fusion technique to improve the performance.

More details and performance of the classifiers are provided in *Section* 3.2.2.

3.2 Experiments and Discussion

In this section, at first, we discuss the database employed for the experiments.

3.2.1 Database Employed

For our experiments, we needed such a database where an individual had written some pages in Bengali and English. Although some open-source handwritten Bengali and English databases were available separately, we did not find any Bengali/English datasets with proper ground-truth stating the writer information as per our requirements. Hence, we generated our own database by inviting 100 volunteers. Those volunteers were native Bengali writers from West Bengal, India, and were of both genders with different ages and academic backgrounds. Each volunteer wrote 3 full pages ($A4$ sized) of Bengali and 3 pages of English from any book/article or from their memory, in their regular handwriting. The $A4$ sized 75 GSM (g/m^2) white pages and black-inked pens with 0.5 mm ball-point of the same brand/model were provided to all writers to obtain prominent ink-strokes of similar color intensity and thickness.

A total of 300 pages of Bengali and 300 pages of English handwriting with writer information were present in our database. A quite realistic scenario was considered, where the labeled dataset (e.g., English) was absent, and we wanted to see the impact of learning by sharing information (e.g., from Bengali script). Here, each page contained about 16 text-lines. Thus, approximately 4,800 (= 300 × 16) Bengali/English text-lines

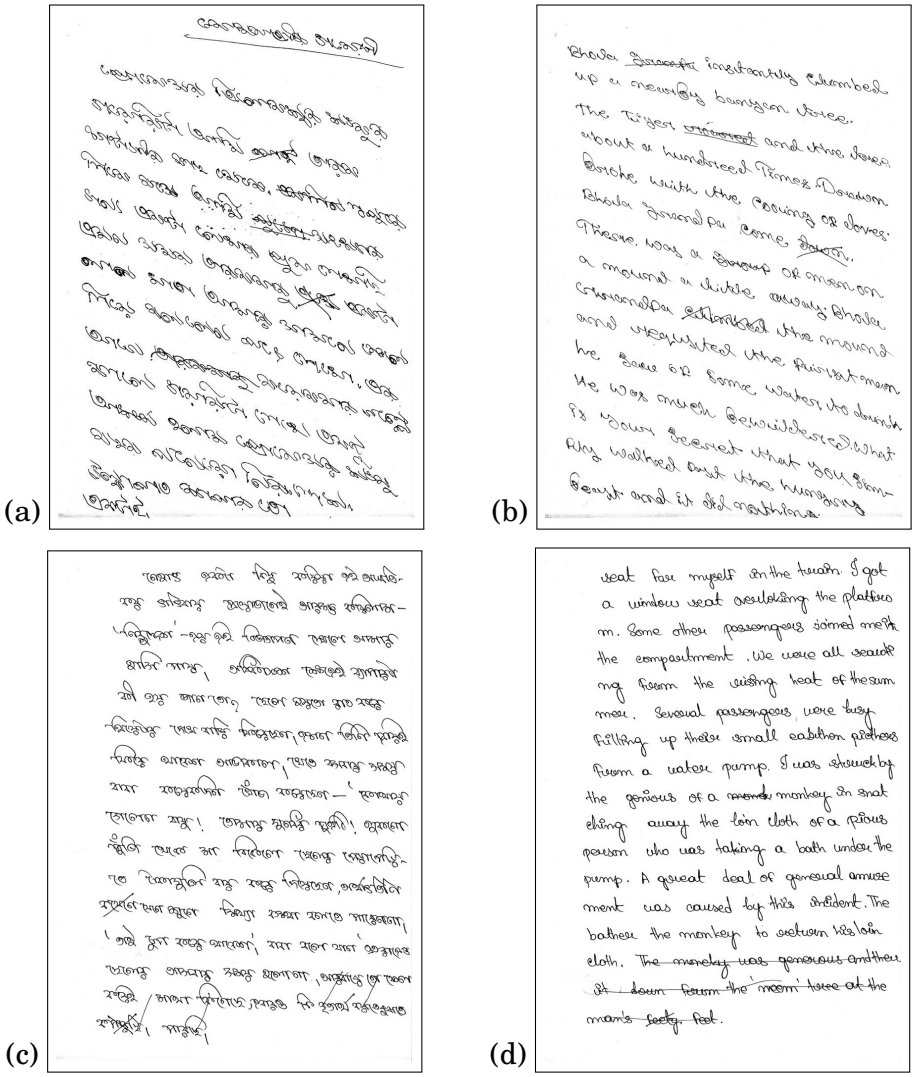


Figure 3.2: Image samples from our database of (a)-(b) Writer-001 and (c)-(d) Writer-064. (a), (c) Bengali and (b), (d) English handwritings.

were present. Typical handwriting specimens from our database are shown in Figure 3.2, where text-line skew variation can be seen.

3.2.2 Results and Evaluation

In this subsection, at first, we discuss the arrangements/settings for the classifiers.

In our kNN classifier, the exhaustive nearest neighbor search with the Euclidean distance function was used. The value of 'k' was set to the square root of the number of data points in the training set.

We used two variants of MLP (say, MLP_1 and MLP_2). For MLP_1, the training function

employed was "scaled conjugate gradient backpropagation". This training function took less memory. Here, the performance function was MSE (Mean Squared Error). Only one hidden layer was used here. The number of neurons in this hidden layer was set empirically from 500 to 100. The transfer function used for activation of neurons was "log-sigmoid". The number of training epochs was increased from 1,000 up to 30,000.

In the MLP_2, we used the "Levenberg-Marquardt back-propagation" algorithm as a training function. It was relatively faster but required more memory. Here also MSE was used as the performance function. Two hidden layers with neurons from 500 to 200 were used here. The "hyperbolic-tangent sigmoid" transfer function was applied to activate the neurons. The count of training epochs was from 1,000 to 30,000.

In the SVM classifier, we used the RBF (Radial Basis Function) as a kernel, where the hyper-parameters (γ and $\mathscr{C}$) were evaluated from a tuning set in the logarithmic scale, for γ in $\{2^{-3}, 2^{-2}, \ldots, 2^{4}\}$ and $\mathscr{C}$ in $\{2^{-2}, 2^{-1}, \ldots, 2^{7}\}$. From experiments in the tuning phase, the best performance was obtained for $\gamma = 2^2$ and $\mathscr{C} = 2^5$, respectively. These values were used throughout the experiments on the test data. For n-class writer identification, we used *one-against-all* SVM, since it works better than the *one-against-one* strategy for handwriting-related tasks [121].

We performed our experiments in two separate experimental setups:

- ***Setup-B/E*** : Training on 300 Bengali pages (3 pages of each writer), and testing on 100 English pages (1 page of each writer).

- ***Setup-E/B*** : Training on 300 English pages (3 pages of each writer), and testing on 100 Bengali pages (1 page of each writer).

For ***Setup-B/E***, we used Bengali handwriting for training. To obtain multiple classifiers (C_1, C_2 and C_3), we trained the classifiers by using 3 subsets (s_1, s_2 and s_3) of the training data. These subsets s_1, s_2, s_3 contained 1, 2, 3 Bengali handwritten pages of each writer, respectively. Therefore, for a total of 100 writers, in s_1, s_2 and s_3, there were approximately 1600, 3200 and 4800 text-lines, respectively. The classifier outputs were combined using the '*sum rule*', i.e., the addition of the confidence measures of the classifiers. This fusion strategy works better than some other fusion schemes such as the *max*, *min*, *median*, *majority vote* and *product rule* [117]. We also performed a fusion of kNN, MLP_1, MLP_2 and SVM classifier outputs to analyze the performance. The test set contained an English handwritten page having approximately 16 text-lines for each writer.

Table 3.1: Top-1 writer identification performance

Setup	*Setup-B/E* (*Setup-E/B*)			
Classifiers	***Accuracy %***			
	C_1	C_2	C_3	$C_1+C_2+C_3$
kNN	55.61 (52.19)	58.27 (54.39)	60.61 (57.55)	60.72 (57.90)
MLP_1	56.58 (52.62)	59.78 (56.17)	63.56 (59.59)	66.69 (62.81)
MLP_2	58.89 (55.07)	62.81 (59.08)	66.84 (63.29)	70.49 (67.38)
SVM	63.72 (60.33)	65.25 (62.07)	68.19 (64.88)	71.90 (68.36)
kNN + MLP_1 + MLP_2 + SVM	63.79 (61.20)	66.25 (62.88)	69.73 (66.26)	**72.81 (69.44)**

Table 3.2: Top-2 writer identification performance

Setup	*Setup-B/E* (*Setup-E/B*)			
Classifiers	***Accuracy %***			
	C_1	C_2	C_3	$C_1+C_2+C_3$
kNN	55.88 (52.64)	60.34 (56.73)	61.22 (58.65)	61.51 (59.10)
MLP_1	58.21 (54.95)	61.04 (57.41)	64.59 (61.03)	67.22 (63.76)
MLP_2	60.82 (57.23)	65.68 (62.39)	67.55 (64.42)	72.22 (68.75)
SVM	65.21 (61.82)	67.74 (64.66)	71.18 (67.71)	73.22 (70.47)
kNN + MLP_1 + MLP_2 + SVM	66.17 (62.44)	68.73 (65.60)	72.14 (68.69)	**74.15 (71.23)**

The ***Setup-E/B*** was just converse of the *Setup-B/E*, where English handwriting was used for training and Bengali for testing. Otherwise, multiple classifiers generation and combination were the same as *Setup-B/E*.

Writer identification and verification tasks were executed on the test set. For writer identification, we used the Top-N criterion, where the possible writer was extracted in a reduced set of 'N' ($\ll$ total number of writers) elements. Here, we chose the Top-1, Top-2 and Top-5 criteria. We present the performance of *Setup-B/E* (*Setup-E/B*) on the basis of *accuracy* in Tables 3.1, 3.2 and 3.3, respectively.

The writer verification performance is shown in terms of *balanced accuracy* (refer to *Section* 2.2.2 of *Chapter* 2). The results of writer verification are shown in Table 3.4. The last row and last column of Tables 3.1 - 3.4 show the results after combining classifiers. It was observed that the combined classifiers consistently outperform the standalone classifiers.

Overall, we obtained better results on *Setup-B/E* than *Setup-E/B*. One possible reason may be that the Bengali script contains more complex character patterns in comparison with English, and training on a complex pattern may provide better results.

Table 3.3: Top-5 writer identification performance

Setup	*Setup-B / E* (*Setup-E / B*)			
Classifiers	***Accuracy %***			
	C_1	C_2	C_3	$C_1+C_2+C_3$
kNN	59.19 (55.62)	61.91 (57.96)	63.08 (59.48)	63.32 (59.88)
MLP_1	60.45 (57.09)	63.52 (59.70)	68.30 (64.98)	69.85 (66.52)
MLP_2	63.17 (59.49)	69.49 (64.76)	70.92 (67.36)	74.68 (70.79)
SVM	67.47 (63.86)	71.50 (67.87)	73.58 (69.71)	76.78 (73.38)
kNN + MLP_1 + MLP_2 + SVM	68.38 (64.62)	71.86 (68.26)	74.76 (71.09)	**77.78 (74.67)**

Table 3.4: Writer verification performance

Setup	*Setup-B / E* (*Setup-E / B*)			
Classifiers	***Balanced Accuracy %***			
	C_1	C_2	C_3	$C_1+C_2+C_3$
kNN	65.62 (62.94)	67.16 (65.12)	71.53 (68.56)	73.25 (70.34)
MLP_1	65.81 (63.05)	68.22 (65.30)	73.07 (70.78)	76.59 (73.94)
MLP_2	69.44 (66.57)	75.42 (72.77)	75.92 (73.16)	80.34 (77.44)
SVM	73.52 (71.42)	76.40 (73.97)	79.59 (77.37)	82.51 (80.13)
kNN + MLP_1 + MLP_2 + SVM	74.64 (71.76)	76.74 (74.85)	80.53 (77.83)	**83.79 (80.96)**

For *comparison* of our research with others, we did not find any work where the training was performed on Bengali (English) handwriting and the testing was undertaken on English (Bengali) data.

Furthermore, we have analyzed some possible reasons behind fairly encouraging outcomes of our work, which are as follows.

(i) Although Bengali and English scripts have no similarity in overall character structures, there are a few shape-wise resemblances between Bengali and English character curvatures, e.g., the English character '**c**' and Bengali grapheme 'ও'. The main difference is that the curvature direction for '**c**' is left-to-right, while for 'ও', it is right-to-left.

(ii) Both English and Bengali text-lines are written in a left-to-right direction. So, there are similarities in writing flow and direction of text-lines.

(iii) India was under British rule for approximately 200 years. For many Bengali writers, their second language is English in school-level education. Therefore, the Bengali writing style may have been influenced by English writing for a long period of time.

Our work may be extended to investigate the impact of writing in various scripts and to study whether there exists any similarity among multiple handwritten manuscripts.

3.3 Summary

In this chapter, we deal with writer identification and verification across scripts using two experimental setups, i.e., *Setup-B/E*: training on Bengali script and testing on English, and *Setup-E/B*: training on English script and testing on Bengali. Here, we use some latent features with multiple classification techniques for writer identification/verification. To evaluate the efficacy of our system, a database of 300 Bengali and 300 English pages, scribbled by 100 writers, has been generated. Writer identification on the Top-1, Top-2 and Top-5 criteria have produced an overall accuracy of 72.81% (69.44%), 74.15% (71.23%) and 77.78% (74.67%), respectively, for *Setup-B/E* (*Setup-E/B*). Also, for writer verification, an 83.79% (80.96%) balanced accuracy has been obtained from *Setup-B/E* (*Setup-E/B*). Our future work will focus on obtaining higher accuracy by introducing some more useful features for discriminating different types of handwriting. Our next endeavor will also be a collaborative study with researchers of various multilingual countries to collect more datasets for further experimental analysis of this work.

CHAPTER 4

WRITER IDENTIFICATION AND VERIFICATION FROM INTRA-VARIABLE WRITING

Handwriting is basically a kind of pattern. However, from the pre-historic era, it bears the connotation of human civilization. The handwriting instrument progressed from finger and wedge (on clay/sand and stone-based medium) to quill, pencil, fountain/ball-point pen (on parchment, papyrus/paper), and again finger (on the touch-screen of a smart device). Though the world is going fast towards a paperless e-world, "handwriting remains just as vital to the enduring saga of civilization (–Michael R. Sull)".

For computer scientists, *automated analysis of handwriting* is a recognized field of study owing to the ever-increasing complexity of extreme variations and having positive impacts on the fields of Forensics, Biometrics, Library Science and Data Science.

As discussed earlier in *Chapter* 1, the handwriting pattern varies with a person due to individual writing style. This may be termed as *inter-class variance*. It is also noted that handwriting samples of a single person may vary extensively with various factors such as mood, time, space (geographical location), writing medium and tool. This is referred to as *intra-class variance*. Sometimes, these inter-class and intra-class variations are termed as "between-writer" and "within-writer" variability, respectively [149]. Even for excessive stroke variation among handwritten specimens of an individual, the writer

This work is partially published as:
C. Adak, B. B. Chaudhuri, M. Blumenstein, "An Empirical Study on Writer Identification & Verification from Intra-variable Individual Handwriting", IEEE Access, vol. 7, no. 1, pp. 24738-24758, 2019.

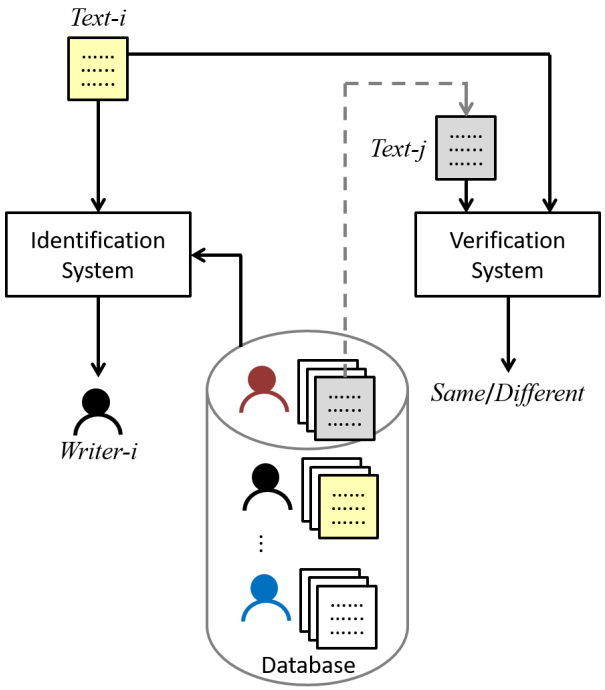


Figure 4.1: Ideal writer identification and verification system.

and others having long exposure to his/her writing may still recognize it. Some implicit stroke characteristics may be the reason behind this ability.

In the field of forensics and biometrics, verifying/identifying a writer from a handwriting sample is sometimes essential (e.g., in the case of the "2001 anthrax attacks"). Nowadays, computer-assisted automated analysis is also quite popular in this application. Writer verification is a task used to authenticate a given document whether it is written by a certain individual or not. In writer identification, the goal is to match the writers to their handwriting specimens. The target of the writer identification/verification task is to maximize inter-variability and to minimize intra-variability.

The workflow of writer identification and verification approach is shown in Figure 4.1. Here, we have a database of handwritten texts with its known authors/writers. A query text sample *Text-i* is input to the writer identification system to obtain its writer-id (*Writer-i*) as an output with a certain degree of accuracy where the database provides support for the retrieval. In the writer verification system, two text samples *Text-i* and *Text-j* are fed to decide whether they are written by the "*Same*" or "*Different*" persons. The *Text-i* is a query sample to be verified and the *Text-j* may be fed from the database of known authors.

In the document image analysis literature, interest has grown in the area of auto-

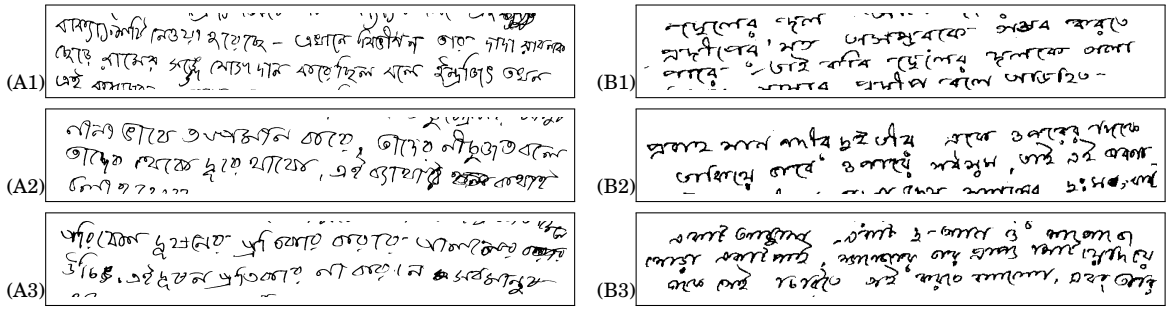


Figure 4.2: Intra-variable Bengali handwritten samples, *left column*: (A1), (A2), (A3) samples are of Writer-A, *right column*: (B1), (B2), (B3) samples are of Writer-B. The intra-variability of Writer-A's samples is less compared to Writer-B's samples, as confirmed by handwriting experts.

mated writer identification/verification for the last four decades. The literature review on this topic is reported in *Section* 1.4.1 of *Chapter* 1. Most of the past research work [186, 204, 252] has focused on ideal handwriting generated in normal circumstances without paying much attention to the intra-variability.

However, a situation may arise where an author needs to be verified based on a quickly written, unadorned handwritten manuscript, whereas only regular neat/clean handwriting with known authorship is available in the training database. Similar situations may occur where we need to identify the writer from unclaimed tidy handwriting, but the available database contains only careless untidy writing. In such a situation, the available writing of the person in the database and the test document written by the same individual can be highly dissimilar. In Figure 4.2, two sets of intra-varied handwritten sample of two individual writers are shown. Here, for example, we may need to verify whether the sample of Figure 4.2:(B1) is written by Writer-B, on the basis of B's handwriting of say Figure 4.2:(B3).

In this chapter, we focus on the situation of intra-variation of individual handwriting to perform writer identification/verification. Ideally, within-writer variation should be less than the between-writer variation, which is the basis of the writer identification/verification task [149]. However, where the intra-variation is relatively higher (refer to Figure 4.2), we need to find some handwriting features less sensitive to intra-variability and more sensitive to inter-variability.

We have generated two sets of database of intra-variable handwriting to deal with such realistic scenarios of writing identification/verification. Our database contains offline handwriting of *Bengali* script which is a fairly complex Indic script [37, 178]. Recent advancements in writer identification on Indic scripts have been reported in

[44]. The general features of Bengali script can be found in [37]. In connection with writer identification, some useful characteristics of Bengali handwriting are mentioned in [44], for example, matra/headline, delta, hole, coil shape, etc. We have noted that these Bengali handwriting characteristics along with classical handwriting characteristics (inter-text-line and inter-word gap, text-line skew, word/character slant, height/width of a character, text main-body height, character formation, etc.) usually vary with writing.

In our generated database, the intra-variation is relatively higher than most of the existing databases in the literature [252], as confirmed by some handwriting experts. Therefore, here, we need to find some handwriting features which can decrease the intra-variability and increase the inter-variability.

The *applicability* of this work is as follows. *First*, it will be helpful in the fields of forensics and biometrics for writer identification and verification. *Second*, this work studies the impact of working with absent data, i.e., when a particular type of individual writing is absent in the training set, how the system performs while testing with that type. *Third*, this work may be useful in some applications of cultural heritage and library science. When some unpublished manuscripts of a scholar are found, the authorship is usually verified [120]. Now, for the case of newly-discovered manuscripts, if they contain some unknown writing styles of the scholar, our work may provide some insights to analyze the authorship. *Fourth*, this work may have a modest understanding of the progress of some diseases such as Parkinson's, Alzheimer's, Dysgraphia, Dyslexia, Tourette syndrome, etc., which affect handwriting. Here, before and after the disease progression, handwritten specimens have high intra-variability and provide some applicability of our work.

In this chapter, we analyze the intra-variable handwriting for writer identification and verification tasks. The writer identification task is perceived as an n-class classification problem to classify handwritten documents in n number of writer classes. On the other hand, writer verification is perceived as a binary classification, where a document is marked with "Same" or "Different" class, if it is written by the *same* or *different* writer of the given document, respectively. For these tasks, we extract two types of features, handcrafted and auto-derived [142], from an offline handwritten text sample set. Then the extracted features are classified to identify/verify a writer. In writer identification, SVM (Support Vector Machine) is used for handcrafted features and some deep neural models are used for auto-derived features. In writer verification, we employ some similarity metrics on handcrafted and auto-derived features.

Our *contribution* to this chapter is the study of writer identification/verification from the intra-variable handwriting of an individual. Such a rigorous investigation is the

earliest attempt of its type. For this study, we have generated two databases containing intra-variable handwriting in a controlled and uncontrolled way. The subgrouping of the uncontrolled database with respect to intra-variability is rather new. Here, the handwritten pages of the uncontrolled database are initially clustered with some deep features, and then finally grouped by confirmation of a classification technique which is pre-trained by the controlled database. We also propose two patch selection tactics to provide the input to the deep architectures without any normalization. Moreover, two writer identification strategies are introduced here, which are relatively new. The data augmentation technique is also a new addition here with respect to the offline handwritten data.

The rest of the chapter is organized as follows. *Section* 4.1 discusses related work and *Section* 4.2 describes the experimental dataset generation procedure. Then *Section* 4.3 mentions the preprocessing step before entering into the methodology. After that *Section* 4.4 and *Section* 4.5 describe the handcrafted and auto-derived feature extraction techniques, respectively. The writer identification procedure is discussed in *Section* 4.6, followed by *Section* 4.7 with a description of the writer verification process. The subsequent *Section* 4.8 is comprised of experimental results and discussion. Finally, *Section* 4.9 summarizes this chapter.

4.1 Related Work on Intra-variable Handwriting

With our both online and offline searching capacities, we have not found any direct work on intra-variable handwriting for writer identification/verification. In this section, we cite some slightly related works mentioning intra-variability in writing.

The handwriting of a person may change for using various writing instruments. The handwriting alteration of a person using a pen and pencil was studied in [119]. Hilton [167] reported that some pens can suppress the writing characteristics of an individual and may introduce intra-variability.

The handwriting of a person may change with the writing surface, more precisely, by the friction between the writing medium (paper) and tool (pen). Gerth et al. [200] studied this within-writer variability when writing was performed on paper and a tablet computer.

Handwriting also changes with time span and the age of a person [127, 133]. Speedy writing in excitement, or writing with a stressed mind may degrade the writing quality, thus produces within-writer variability [22, 133].

Some diseases such as Parkinson's, Alzheimer's, etc. affect the handwriting of an individual [110, 127]. Therefore, before and after such diseases, the writing shows intra-variability.

Alcohol consumption also changes the individual's writing style and provides an example of intra-variable handwriting [79].

Such *mechanical* (e.g., writing instrument, surface, etc.), *physical* (e.g., illness, aging, etc.), *psychological* (e.g., excitement, anger, mood, etc.) factors may cause large individual handwriting variation [133].

4.2 Experimental Dataset

For our experimental analysis, we needed a database of intra-variable handwritten samples of each writer. We did not find any such publicly available database. Therefore, we had to generate a new database containing such handwriting specimens. We made two offline databases on the basis of a handwriting collection strategy, namely *controlled* and *uncontrolled* databases.

4.2.1 Controlled Database (D_c)

In this case, all volunteers chosen for supplying data were aware of the modality of our experiments. For controlled database collection, we focused on intra-variability occurring mostly due to writing speed [128]. It is noted that typically speedy writing is more distorted than the usual handwriting of a writer, and this inference was confirmed by some handwriting experts.

Collecting previously written samples at various speeds from many writers is quite challenging, since no one typically monitors and records his/her writing speed unless required for a specific reason. Therefore, we collected handwriting under a controlled setup, as follows.

All volunteers were instructed to write at various speeds. At first, they were advised to write at their normal speed of writing. Then they were instructed to write faster than their normal writing speed. Finally, they were requested to write at a slower speed than normal. We labeled these as *medium* (or, normal), *fast*, and *slow* sets of handwriting samples, respectively.

At the stage of full-page writing, we noted the total time (t) of writing using a stop-watch and computed the total length (l) of writing strokes on a page. The handwriting

speed (s) in a full page is calculated as stroke length per unit time, i.e., $s = l/t$. The stroke length is computed from counting the number of object pixels from the thinned version of the writing strokes. For thinning purpose, the *Zhang-Suen thinning* method [225] was used which worked better than some other techniques [140]. Additionally, in the thinned version, a spurious branch having length less than half of the average stroke-width is pruned.

The reaction time ($\approx$ 200 milliseconds) [21] for using a stop-watch is negligible with respect to the objective of our task. We prepared our experimental setup to write on a white 70 GSM (g/m^2) A4 page and placing it on a horizontal plane surface with a smooth ball-point pen having black/blue ink. We created our database in offline mode, and did not use any digital pen/surfaces which could obstruct the individual writing habits. Here, our primary aim is to capture the intra-variability of handwriting, on which our technique can work adequately without using accurate, ultramodern speed measuring instruments.

For a writer W_i, a fast handwritten page is chosen where handwriting speed (s_i, a scalar quantity) is greater than a threshold $T1_i$, i.e., $s_i > T1_i$. For slow handwriting, $s_i < T2_i$, where $T2_i$ is another threshold. The medium handwritten page is chosen where $T1_i \geq s_i \geq T2_i$. The medium or normal handwriting speed is captured at first from multiple handwritten pages of a writer W_i. We calculate mean (μ_{si}) and standard deviation (σ_{si}) of medium speed from these pages. Here, we use $T1_i = \lceil \mu_{si} + \alpha_s.\sigma_{si} \rceil$ and $T2_i = \lceil \mu_{si} - \alpha_s.\sigma_{si} \rceil$. We set $\alpha_s = 2$, which is decided empirically.

Each writer wrote multiple pages which were ordered in terms of the writing speed. We chose top speedy two pages from the *fast* handwritten samples, two lowest-speeded pages from the *slow* writing, and middle-speeded two pages from the *medium* writing.

Although many volunteers contributed to our database generation, we chose 100 writers whose handwriting patterns varied structurally due to writing speeds, as advised by some handwriting experts. This is performed in order to generate a database of intra-variable handwriting.

The writers were native Bengali from West Bengal, India. The finally selected 100 writers were in the age group of 12-42 years having academic backgrounds from secondary school to university level. The ratio of male to female writers in this database is 14:11.

This controlled dataset (D_c) contains 600 pages where each writer contributed 6 pages of Bengali handwriting. Among 6 pages, 2 pages are of *fast* handwriting speed, 2 pages are of *medium* and the remaining 2 pages are of *slow* speed. In other words, we have 3 sets of handwriting namely S_f (*fast*), S_m (*medium*), S_s (*slow*), each containing 2

handwritten pages of every 100 writers.

4.2.2 Uncontrolled Database (D_{uc})

For the generation of uncontrolled database, the writers should be unaware of our experiments before the data collection. However, people generally perform their daily writing at a normal pace using various pens and papers, and a uniform data collection setup is missing. Here, we came up with a different strategy for this type of database generation.

After discussion with some handwriting experts, we note that in real-life school examinations, the students generally write at various speeds, since the examination time is limited and the usual target is to score good marks by answering all questions within that stipulated period. Therefore, the students/writers are in a hurry when the clock is ticking towards the end of the exam. However, here some behavioral/psychological aspects [25, 128] may influence intra-variable handwriting, besides the writing-speed, due to anxiety, nervousness to finish, the panic of a low score, stress to recollect the answer, etc.

Therefore, we have selected a real school-exam scenario to collect suitable data by maintaining uniformity for this data collection. Some details of such a scenario are as follows.

(i) Time: The examination duration was fixed as three hours. Here, the time works as a constraint to ascertain the increase/decrease of writing speed and individual variability.

(ii) Question type: We selected a 100-mark Bengali literature examination paper, where most of the question types were broad subjective to be answered in many sentences.

(iii) Script: The answers were to be written in Bengali script.

(iv) Paper: For writing the answers, white pages of 70 GSM (g/m^2) with a fixed size of 215.9 × 355.6 mm^2 were provided.

(v) Pen: The writers used their own pens. Most of the pens were of black/blue ink with 0.5 - 1.0 mm ball-point tip.

(vi) Writer: The writers were native Bengali from West Bengal, India, and students of VIII - XII grade Bengali-medium public schools. Their age ranged between 13 - 19 years. All these writers were different from the volunteers participated in controlled database generation.

The exam marking was performed by school teachers. For our task, we chose the answer script of a student who scored at least 40% and wrote at least 6 full pages. A total of 153 writers were chosen in this way. Among these writers, 67 persons wrote 6

pages each, 11 persons wrote 7 pages each, 54 persons 8 pages each, 21 persons 9 pages each. Therefore, 153 writers contributed a total of 1100 pages.

Now, we have 1100 unlabeled pages of 153 writers. We label each page in one of the 3 groups of intra-variable writings, say, S'_{fa}, S'_{ma} and S'_{sa}. For this grouping, at first, we use auto-derived feature-based clustering techniques, as follows.

We use the front part of GoogLeNet [63] for feature extraction, since this architecture provides encouraging accuracy with a comparatively lesser dimension of feature vector [160]. From each page, an n_p number of text-patches of size 224 × 224 is chosen arbitrarily. This n_p is set empirically to 400. For each patch, we have obtained 1024-dimensional deep feature vector from the *avg pool* layer of GoogLeNet [63]. For clustering, we choose various clustering algorithms such as K-means, Fuzzy C-means, Minibatch K-means, Expectation-Maximization with Gaussian-Mixture-Models (EM with GMM), and Agglomerative Hierarchical Clustering (Agglo_Hierarchical) [10].

Let us assume that p_{ij} denotes $\mathrm{patch}_{i \mid i=1,2,\ldots,400}$ of $\mathrm{page}_{j \mid j=1,2,\ldots,1100}$. Actually, p_{ij} represents a 1024-dimensional feature vector obtained from patch_i of page_j. For p_{1j}, we obtain a cluster plot CL_1 by patch_1's of all the 1100 pages. Similarly, cluster plot CL_2 is obtained from patch_2's of all 1100 pages. And so on, patch_{400}'s of all 1100 pages produce cluster plot CL_{400}. These 400 cluster plots are ordered with a corresponding patch; i.e., $\mathrm{patch}_\mathrm{i}$'s (or, the i^{th} patch) of all pages produce CL_i. Moreover, we obtain 600 (fixed empirically) cluster plots unordered with patches which are chosen arbitrarily. Therefore, we now have 1000 (= 400 + 600) cluster plots. Each cluster plot contains 1100 patches (i.e., patch-based feature points), where each patch represents each of the 1100 pages. From these 1000 cluster plots, using the *majority rule*, we label each page into three groups/clusters S'_{fa}, S'_{ma} and S'_{sa}. For example, suppose in all the 1000 cluster plots, the majority of the patches of page_1 (p_{i1}) fall in the cluster S'_{fa}; then the page_1 is put into the group S'_{fa}.

To find which clustering algorithm among K-means, Fuzzy C-means, Minibatch K-means, EM with GMM, and Agglo_Hierarchical, would work well for intra-variable handwriting, we use an external evaluation criterion, called NMI (*Normalized Mutual Information*) score [51]. For this, we perform a similar feature extraction strategy and clustering techniques on the controlled dataset D_c, which contains the ground-truth. Employing various clustering techniques on D_c, we have obtained the NMI scores, as presented in Table 4.1. The Agglo_Hierarchical method worked well on D_c, so we use this clustering technique here also on uncontrolled data.

Up to this point, the 1100 unlabeled pages of 153 writers are clustered into 3 groups

Table 4.1: Clustering method evaluation on D_c

Clustering Method	***NMI Score***
K-means	0.637
Minibatch K-means	0.610
Fuzzy C-means	0.646
EM with GMM	0.679
Agglo_Hierarchical	**0.684**

(S'_{fa}, S'_{ma} and S'_{sa}) using an unsupervised clustering technique (Agglo_Hierarchical).

Now, we classify the 1100 pages into 3 classes (S'_{fb}, S'_{mb} and S'_{sb}) supervised by the controlled database D_c. For this classification, we use GoogLeNet due to its promising performance in other computer vision related tasks [63]. Here also, we arbitrarily choose n_p (= 400, fixed empirically) number of text patches in a page and input to the GoogLeNet. A page is classified into a certain class where the majority of its patches fall.

To assess the efficiency of the GoogLeNet on intra-variable handwriting, we perform a 3-class classification experiment on D_c to classify in S_f, S_m, S_s sets/classes due to having the appropriate ground-truth. For this, we divide D_c into training, validation, and test sets in the ratio of 2:1:1. The performance on the test set of D_c for this 3-class classification problem is 96.87%, which is quite satisfactory for our task.

For classification of uncontrolled handwritten samples (1100 pages), we perform training on the entire controlled database D_c and test on these 1100 pages. After this classification, we have obtained the members of 3-classes S'_{fb}, S'_{mb} and S'_{sb}.

Our intention is to generate 3 sets (S'_f, S'_m and S'_s) of samples containing intra-variable handwriting of an individual. Therefore, each of these 3 sets must contain handwriting samples of every writer.

From the unsupervised clustering, we obtain S'_{fa}, S'_{ma} and S'_{sa} sets. From supervised classification, we get the sets S'_{fb}, S'_{mb} and S'_{sb}. From 153 writers, we choose a certain writer who has at least 2 handwriting samples in each of the S'_{fa}, S'_{ma}, S'_{sa} sets and the same 2 samples in each of the corresponding S'_{fb}, S'_{mb}, S'_{sb} sets. Finally, these 3×2 samples of a writer are put in S'_f, S'_m and S'_s sets, respectively. Out of 153 writers, this constraint is successfully satisfied by 104 writers. Furthermore, we take advice from some handwriting experts and finally choose 100 writers from the 104 writers.

Our uncontrolled database (D_{uc}) contains 3 sets (S'_f, S'_m and S'_s) of intra-variable handwriting samples of 100 writers. Each of the S'_f, S'_m and S'_s sets contain 2 samples per writer. Therefore, similar to the controlled database D_c, this uncontrolled database D_{uc} also contains 600 pages in total. The ratio of male to female writers in D_{uc} is 31:19.

In this uncontrolled database, we observe that the handwriting of most students becomes structurally distorted and unadorned in the latter pages of the answer booklet.

4.3 Preprocessing

All the handwritten pages were scanned by a flat-bed scanner at 300 ppi (*pixels per inch*) in 256 gray-values to obtain digital document images. In the preprocessing stage, we label the components of a handwritten document image using a relatively faster single-pass connected component labeling algorithm [87]. The text region is extracted after removal of the non-text components if present any, using the method of [42]. In the text region, the struck-out texts are also deleted by employing the method of [36], since the presence of struck-out text impedes the usual writer identification performance [48]. However, the style of strike-out strokes [36] may be utilized for writer inspection, which is out of the scope of our current work. Very small sized components such as dots, dashes, commas, colons, etc., and noise are also removed. The text-lines and words are segmented using an off-the-shelf 2D Gaussian filter-based method *GOLESTAN-a*, as discussed in [166]. Character level segmentation is also performed using a water reservoir principle-based method [228].

4.4 Handcrafted Feature Extraction

A writer identification task can be viewed as a multi-class classification problem, where the task is to assign the writer-id to the unknown handwritten specimens. Similarly, writer verification can be perceived as a binary classification problem where the task is to answer *yes*/*no* to a questioned handwritten sample as to whether it has been written by a particular writer. The features used for these tasks are described in this section and the following section. We employ both handcrafted features and auto-derived features [142]. Handcrafted features are required to be predesigned explicitly in the traditional way, whereas auto-derived features do not have any explicit design.

The extracted handcrafted features are discussed as follows.

4.4.1 Macro-Micro Features (F_{MM})

The macro and micro features of Srihari et al. [216] are quite popular since those were very effective in writer identification from handwritings of 1500 U.S. population having

various ethnic groups/ages/genders. Here, we adapt this set of features for our task.

Initially, we choose the macro feature vector, described in [216], which contains 11 features: gray-level entropy (f_1), gray-level threshold (f_2), count of black pixels (f_3), interior/exterior contour connectivity (f_4-f_5), vertical/negative/positive/horizontal contour slope (f_6-f_9), average slant and height of the text-line (f_{10}-f_{11}). From [216], we note that the features f_1, f_2, and f_3 are related to the pen pressure. Here, f_4 and f_5 reveal the writing movement. The features f_6, f_7, f_8, and f_9 are related to stroke formation. The feature f_{10} represents the writing slant and f_{11} is related to the text proportion.

Two paragraph-level macro features are also considered: height to width ratio of a paragraph, i.e., aspect ratio (f_{12}) and margin width (f_{13}). Three more word-level macro features are also employed, which are upper zone ratio (f_{14}), lower zone ratio (f_{15}) and length (f_{16}). We calculate these paragraph and word-level features over a page and take the average value.

The character-level micro features contain 192-bit gradient, 192-bit structural and 128-bit concavity features, concatenated into a 512-bit feature. The detailed description of these features can be found in [216]. We modify this micro feature slightly to get a page-level feature vector. From a page, we obtain the histogram of this 512-bit feature and normalize it by the character count.

The macro and micro features are concatenated to generate the feature vector F_{MM}.

4.4.2 Contour Direction and Hinge Features (F_{DH}):

For writer identification, stroke direction and curvature-based features have been reported to work well [186, 252]. Therefore, we use here the famous contour direction and hinge distribution of handwritten strokes proposed by Bulacu and Schomaker [149].

Along the writing stroke contour, an angle (ϕ) histogram is generated and normalized into a probability distribution $p_f(\phi)$. From the horizontal direction, the angle (ϕ) is calculated as:

$$\phi = tan^{-1}\frac{y_{i+\epsilon} - y_i}{x_{i+\epsilon} - x_i}; \tag{4.1}$$

where, x_i and y_i denote the row and column indices of the i^{th} object pixel. The ϵ depends on stroke-thickness and is fixed as 5 in [149]. In our task, ϵ (> 1) is data-driven, and worked well for $\epsilon = \max(2, \lfloor \mu_{sw} - \sigma_{sw} \rfloor)$, where μ_{sw} is the average stroke-width and σ_{sw} is the standard deviation of stroke-width in a page. The number of histogram bins (n_b) is set as 12 within the range of $0^o - 180^o$. Hence, 15^o per bin is engaged. Clearly, the dimension of this feature $p_f(\phi)$ or f_{cd} is 12.

In [149], for the contour hinge feature f_{ch}, two contour fragments, joined to a common end, making angles ϕ_1 and ϕ_2 (where, $\phi_2 \geq \phi_1$), spanning all four quadrants (360^o), are considered. A normalized histogram is generated with a joint probability distribution $p_f(\phi_1, \phi_2)$. Similar to [149], the number of histogram bins (n_b) is set to 12, leading to $n_b(2n_b + 1) = 300$-dimensional feature vector.

By concatenating f_{cd} and f_{ch}, we obtain F_{DH}.

4.4.3 Direction and Curvature Features at Keypoints (F_{DC})

We intend to ascertain some similarities between handwriting specimens of an individual. Therefore, we focus on some points of interest, i.e., *keypoints* (ρ_i), on the handwritten strokes. These keypoints are obtained by combining some structural points (i.e., start/end, branch, and curved points) and SIFT (Scale-Invariant Feature Transform) keypoints on Bengali handwritten ink-strokes, as described in [44].

Here, our plan is to observe the movement of writing strokes on these keypoints, and therefore, we capture the stroke direction and curvature at the keypoints. For *direction* and *curvature* feature extraction from *offline* handwritten strokes, we use the idea of "The NPen++ Recognizer" [209] which deals with *online* handwriting.

We calculate the writing *direction* between two connected keypoints ρ_i and ρ_{i+1} in terms of Cosine and Sine values and use them as features f_{dc} and f_{ds}, respectively.

$$f_{dc} \equiv cos(\theta_i) = \frac{\rho_{i+1}.x - \rho_i.x}{d_i} \tag{4.2}$$

$$f_{ds} \equiv sin(\theta_i) = \frac{\rho_{i+1}.y - \rho_i.y}{d_i} \tag{4.3}$$

where, $\rho_i.x$ and $\rho_i.y$ are the row and column indices of ρ_i, and
$d_i = \sqrt{(\rho_{i+1}.x - \rho_i.x)^2 + (\rho_{i+1}.y - \rho_i.y)^2}$.

The *curvature* of a writing stroke is the angle made by the line fragments $\overline{\rho_{i-1}\ \rho_i}$ (from ρ_{i-1} to ρ_i) and $\overline{\rho_i\ \rho_{i+1}}$ (from ρ_i to ρ_{i+1}). The Cosine and Sine values of this angle are calculated and employed as features f_{cc} and f_{cs}, respectively.

$$f_{cc} \equiv cos(\theta_i - \theta_{i-1}) = cos\theta_i cos\theta_{i-1} + sin\theta_i sin\theta_{i-1} \tag{4.4}$$

$$f_{cs} \equiv sin(\theta_i - \theta_{i-1}) = sin\theta_i cos\theta_{i-1} - cos\theta_i sin\theta_{i-1} \tag{4.5}$$

For each of these four features (f_{dc}, f_{ds}, f_{cc}, f_{cs}), we generate separate normalized histograms spanning the range of [-1, 1] for a number of bins $n_b = 200$. Therefore, the dimension of each feature vector is 200.

Concatenating features f_{dc}, f_{ds}, f_{cc} and f_{cs}, we get F_{DC}.

4.5 Auto-derived Feature Extraction

The auto-derived features are mainly extracted using a convolutional neural network (CNN). The convolutional architecture generally contains two parts: front and rear. The front part typically extracts the features. The rear part is used for classification (refer to *Section* 4.6.2).

The front part of the CNN takes an input image. We use some patch-based strategies to feed fixed sized input images [235]. Here, we do not use any image normalization, since it impedes the writer identification performance [29]. The patch selection is not performed through the classical sliding-window technique, since the text-lines are not skew-normalized. Sliding a window horizontally through the middle of the text-line (main text-body height) is conceivable, but the information may be lost for several cases such as for highly skewed text-lines, for overlapping text-lines with lesser inter-text-line gaps, etc.

Here, two types of patches are selected as follows.

(a) $\boldsymbol{patch_{char}}$: We already have the character-level information from the pre-processing stage. We find the center of gravity (p_{CG}) of a segmented character image. Then we take a $n_{char} \times n_{char}$ window centering the p_{CG}, and consider it as a character-level patch, say $\text{patch}_{\text{char}}$.

(b) $\boldsymbol{patch_{allo}}$: We have obtained some keypoints on writing strokes, as mentioned in *Section* 4.4.3. A neighboring window of size $n_{allo} \times n_{allo}$ centered at a keypoint is used as a patch. This patch is an allographic-level patch, say $\text{patch}_{\text{allo}}$.

From a text sample, all the $\text{patch}_{\text{char}}$s and $\text{patch}_{\text{allo}}$s are extracted. Each $\text{patch}_{\text{char}}$ is fed to the front part of the CNN and a feature vector f_{pc} is obtained. Similarly, for each $\text{patch}_{\text{allo}}$, a feature vector f_{pa} is generated.

In our task, the following deep-learning architectures are used separately for $\text{patch}_{\text{char}}$ and $\text{patch}_{\text{allo}}$ as inputs.

4.5.1 Basic_CNN

The LeNet-5 is a celebrated convolutional network that works well on the various machine learning problems [254]. Our Basic_CNN architecture is primarily influenced by this LeNet-5 and provides an initial flavor of a deep learning model for our task. Here, we use 3 convolutional layers (C_i), each followed by a sub-sampling (max-pooling, MP_i) layer. The used feature map count with map size, filter size, stride (s), padding (p) values are shown in Figure 4.3. For example, the first convolutional layer (C_1) contains 8 feature

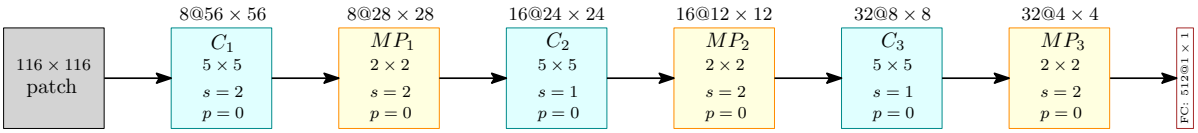


Figure 4.3: $\text{BCNN}_{\text{char}}$: Basic_CNN architecture as a feature extractor while using $\text{patch}_{\text{char}}$.

maps of size 56×56 each, and each feature map is connected to a 5×5 neighbor window of the input. Here, $s = 2$ and $p = 0$ is used. For each convolutional layer, instead of the *tanh* activation function of LeNet-5, we use ReLU (Rectified Linear Unit) [242] due to its advantages of sparsity and reduced likelihood of vanishing gradient.

For feature extraction using both $\text{patch}_{\text{char}}$ and $\text{patch}_{\text{allo}}$, the used Basic_CNN architectures ($\text{BCNN}_{\text{char}}$ and $\text{BCNN}_{\text{allo}}$) are almost similar except some minor differences. The $\text{patch}_{\text{char}}$ size is $n_{char} \times n_{char}$, and $\text{patch}_{\text{allo}}$ size is $n_{allo} \times n_{allo}$. Here, $n_{allo} = \lfloor n_{char}/2 \rfloor$ is used. We fix the n_{char} as 116. For feeding $\text{patch}_{\text{char}}$ in the first convolutional layer (C_1) of $\text{BCNN}_{\text{char}}$, $s = 2$ and $p = 0$ are used. However, for C_1 of $\text{BCNN}_{\text{allo}}$, $s = 1$ and $p = 1$ are employed. The rest of the $\text{BCNN}_{\text{allo}}$ architecture is kept similar to $\text{BCNN}_{\text{char}}$ (refer to Figure 4.3). In this case, we use SGD (Stochastic Gradient Descent) as optimizer with *initial_learning_rate* = 0.01, *momentum* = 0.9, and *weight_decay* = 0.0005. The other parameters of Basic_CNN are similar to the LeNet-5 [254].

Employing this Basic_CNN, we obtain a feature vector of size 512 for each patch.

4.5.2 SqueezeNet

The AlexNet [20] is one of the pioneering models of deep learning revolution and the winner of ILSVRC (ImageNet Large Scale Visual Recognition Challenge)-2012 [169]. The recent deep learning era has been started from AlexNet [160]. For our task, we employ some major deep learning architectures, discussed here and the following subsections.

The SqueezeNet architecture provides AlexNet-level accuracy with lesser parameters and reduced demand on memory [83]. Therefore, we use SqueezeNet here, instead of AlexNet. The details of the SqueezeNet can be found in [83], and we use the *Simple Bypass* version of this network. Here, we call the layers by their names as used in [83]. For our task, we use the same weights trained on ImageNet [20, 169] by adapting the concept of *transfer learning* [208].

We select $n_{char} \times n_{char}$ sized $\text{patch}_{\text{char}}$ and $n_{allo} \times n_{allo}$ sized $\text{patch}_{\text{allo}}$ to be fed separately to the SqueezeNets (SN_{char} and SN_{allo}), where $n_{allo} = \lfloor n_{char}/2 \rfloor$. The SqueezeNet

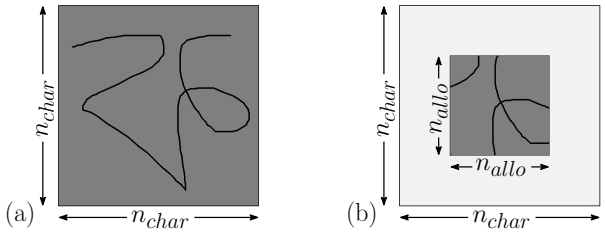


Figure 4.4: (a) $\text{patch}_{\text{char}}$ of size $n_{char} \times n_{char}$, (b) $\text{patch}_{\text{allo}}$ of size $n_{allo} \times n_{allo}$ is shown in dark-gray, and the zero-padding, bounding the $\text{patch}_{\text{allo}}$ is shown in light-gray color.

takes a standard input of fixed size, i.e., 224 × 224. Here, n_{char} equals to 224 and consequently, n_{allo} becomes 112. Therefore, we use a zero-padding of width 56 (= $\lfloor (n_{char} - n_{allo})/2 \rfloor$) to the boundary of $\text{patch}_{\text{allo}}$, as shown in Figure 4.4, for maintaining the standard input size of SqueezeNet.

After the *conv10* and *avgpool10* (refer to [83]) layer of SqueezeNet, we obtain a 100 (number of writers)-sized feature vector for each patch.

4.5.3 GoogLeNet

We choose this network, since it won the ILSVRC-2014 [169] competition and obtained a performance closer to human-being. The details of this architecture can be found in [63]. This GoogLeNet architecture is also called as *Inception V1*. We employ two separate GoogLeNets (GN_{char} and GN_{allo}) to feed $\text{patch}_{\text{char}}$ and $\text{patch}_{\text{allo}}$. The input patch size is similar to the SqueezeNet. The size of $\text{patch}_{\text{char}}$ is 224×224 and $\text{patch}_{\text{allo}}$ is 112×112. Here also, $\text{patch}_{\text{allo}}$ is bound by a zero-padding of width 56. The rest of the GN_{char} and GN_{allo} architectures are similar to the GoogLeNet of [63]. The weights are transferred by pre-training of GoogLeNet using ImageNet data.

After the *avg pool* layer [63], a 1024-dimensional feature vector can be obtained.

4.5.4 Xception Net

The refined versions of GoogLeNet (*Inception V1*) [63] are *Inception V2* [207] and *Inception V3* [62]. Also, the Xception Net [80] is a stronger version of the *Inception V3*. Therefore, we use this Xception Net (with two fully connected layers). The name "Xception" is coined from "Extreme Inception".

The $\text{patch}_{\text{char}}$ fed Xception Net (XN_{char}) takes $n_{char} \times n_{char}$ sized input images and follows the architecture of [80]. Xception Net [80] takes 299×299 sized input image. Therefore, we use $n_{char} = 299$. A separate Xception Net (XN_{allo}) is used to feed $\text{patch}_{\text{allo}}$

of size $n_{allo} \times n_{allo}$. Here, we use $n_{allo} = \lfloor n_{char}/2 \rfloor = 149$. Since, Xception Net takes fixed sized input of 299 × 299, a zero-padding of width 75 ($= \lfloor (n_{char} - n_{allo})/2 \rfloor$) is employed here, similar to the scheme used for SqueezeNet (refer to *Section* 4.5.2, Figure 4.4). The pre-trained Xception Net on ImageNet data by the transfer learning [80] is used here with the same weights.

We obtain a 2048-dimensional feature vector from the *GlobalAveragePooling* layer [80] of both XN_{char} and XN_{allo}.

4.5.5 VGG-16

The VGG architecture was the runner-up of the competition ILSVRC-2014. We choose the 16 layers' VGG architecture due to its simplicity and uniformity in convolutions. The detail of this architecture is reported in [136].

We use two VGG-16 networks (VN_{char} and VN_{allo}). The VGG-16 takes a fixed size input of 224 × 224. Therefore, we input 224 × 224 sized $\text{patch}_{\text{char}}$ to the VN_{char}. Similar to the SqueezeNet, here also, we use 112 × 112 sized $\text{patch}_{\text{allo}}$ with 56 pixels wide zero-padding to feed to the VN_{allo}. Otherwise, VN_{allo} and VN_{char} networks are the same, and follow the architecture of VGG-16 pre-trained on ImageNet data [136].

After the *FC-4096* layer [136], we obtain a 4096-dimensional feature vector from each of VN_{allo} and VN_{char}.

4.5.6 ResNet-101

This architecture won ILSVRC-2015 and beat human-level performance on ImageNet data [169]. Although ResNet is very deep, it is faster and has fewer parameters compared to the VGG network. The novelty of the ResNet (Residual Network) is its residual or skip connections. The details of this architecture can be found in [132] and we use the ResNet with 101 layers. Here, we use two such nets (say, RN_{char} and RN_{allo}).

The ResNet also takes fixed sized, i.e., 224 × 224 input. Therefore, we feed 224 × 224 sized $\text{patch}_{\text{char}}$ as input to the RN_{char}. Similar to the SqueezeNet, we feed 112 × 112 sized $\text{patch}_{\text{allo}}$ with zero-padding of a width of 56 to the RN_{allo} (refer to *Section* 4.5.2). The rest of the RN_{allo} is similar to the RN_{char}, and both of them follow the architecture of ResNet-101 as reported in [132]. The weights are transferred by pre-training on ImageNet database [132, 169].

For each of the RN_{char} and RN_{allo}, we obtain a 2048-dimensional feature vector after the *avg pool* layer [132].

4.6 Writer Identification

As discussed earlier in this chapter, the writer identification problem is a multi-class classification problem, where the number of classes is equal to the total count of writers.

4.6.1 Handcrafted Feature-based Identification

The handcrafted feature vector obtained from a text sample is fed to an SVM classifier to mark the text sample to its writer-id. The SVM generally works well for multi-class classification in a wide range of pattern recognition applications [255]. With regards to the SVM-based multi-class classification for handwriting-related tasks, the *one-against-all* strategy works better than the *one-against-one* [121]. It can also be noted from [36, 57] that the SVM with an RBF (Radial Basis Function) kernel [241] works better than some other classifiers such as kNN (k-Nearest Neighbors), MLP (Multi-Layer Perceptron), MQDF (Modified Quadratic Discriminant Function) and SVM-linear for *Abjad* (Farsi), *Alphabetic* (English) and *Abugida* (Bengali) handwriting. Hence, we use the one-against-all SVM-RBF for our task.

The SVM-RBF hyper-parameters $\mathscr{C}$ and γ are essential to be tuned to avoid overfitting and to regulate the decision boundary, respectively [129]. For optimal performance of the classifier, the hyper-parameters are selected from a tuning set. We use the traditional grid-searching technique for this purpose [241]. A suitable value for $\mathscr{C}$ is chosen at first from a range of values by cross-validation and then several γ's are tested from a range of values for better $\mathscr{C}$'s.

The best performance is obtained for $\mathscr{C} = 2^2$ within the range $[2^{-3}, 2^{-2}, \ldots, 2^6]$ and $\gamma = 2^4$ within the range $[2^{-3}, 2^{-2}, \ldots, 2^8]$. Here, 5-fold cross-validation is used.

4.6.2 Auto-derived Feature-based Identification

From the text samples, writers are classified using the rear part of the convolutional neural architectures.

For Basic_CNN, the rear classifier part is actually an MLP with 1 hidden layer containing 256 nodes, set empirically. The output layer's nodes depict the number of writer classes.

The rear part of SqueezeNet, GoogleNet, Xception Net, VGG-16, ResNet-101 are used for writer identification (classification). The rear parts of the SqueezeNet, GoogleNet, Xception Net, VGG-16, ResNet-101 commence after the *avgpool10* [83], *avg pool* [63],

GlobalAveragePooling [80], *FC-4096* [136], *avg pool* [132] layers, respectively. All these respective rear part classifiers follow their original architecture [63, 80, 83, 132, 136].

We have obtained the features from multiple patches of a handwritten page. Now, to identify a writer on a whole page, the following two strategies are used. The inputs of the classifiers are also based on the following two strategies.

(a) *Strategy-Major* : On the basis of the feature vector (f_{pi}) extracted from each patch (p_i), we classify the writer (W_{pi}) individually on each patch. In other words, we label each of the multiple patches of a page with a writer-id.

Next, we apply *majority rule* to find the ultimate writer (W) of the page. For example, on a page, if the majority of the text patches are marked with writer-A, then the overall page is considered as written by writer-A. This strategy is presented in Algorithm 1.

Algorithm 1 *Strategy-Major*

Input: $f_{p1}, f_{p2}, \ldots, f_{pn}$| feature vectors of patches $p_1, p_2, \ldots, p_n$ in a page;
Output: W | writer of the page;
for $i = 1, 2, \ldots, n$ **do** ▷ n := number of patches
    W_{pi} = classify(f_{pi}); ▷ W_{pi} := writer of patch p_i
end for
W= majority (W_{p1}, W_{p2}, …, W_{pn});

(b) *Strategy-Mean* : The individual feature vector (f_{pi}) obtained from each of the patches (p_i) of a page is extracted. Here, we calculate the arithmetic mean (m_{pi}) of a feature vector (f_{pi}) obtained from each patch (p_i). Therefore, for each patch (p_i), we have a single scalar mean value (m_{pi}). From the mean values of all patches, we generate a mean feature vector (m_p) for a page. This mean feature vector is used to classify a page into writer class (W). In Algorithm 2, we present this strategy.

Algorithm 2 *Strategy-Mean*

Input: $f_{p1}, f_{p2}, \ldots, f_{pn}$| feature vectors of patches $p_1, p_2, \ldots, p_n$ in a page;
Output: W | writer of the page;
for $i = 1, 2, \ldots, n$ **do** ▷ n := number of patches
    m_{pi} = arithmeticMean(f_{pi});
end for
$m_p = \{m_{p1}, m_{p2}, \ldots, m_{pn}\}$; ▷ m_p = feature vector of m_{pi}'s
W = classify (m_p);

For easy and quick understanding, *Strategy-Major* and *Strategy-Mean* are diagrammatically represented in Figure 4.5.

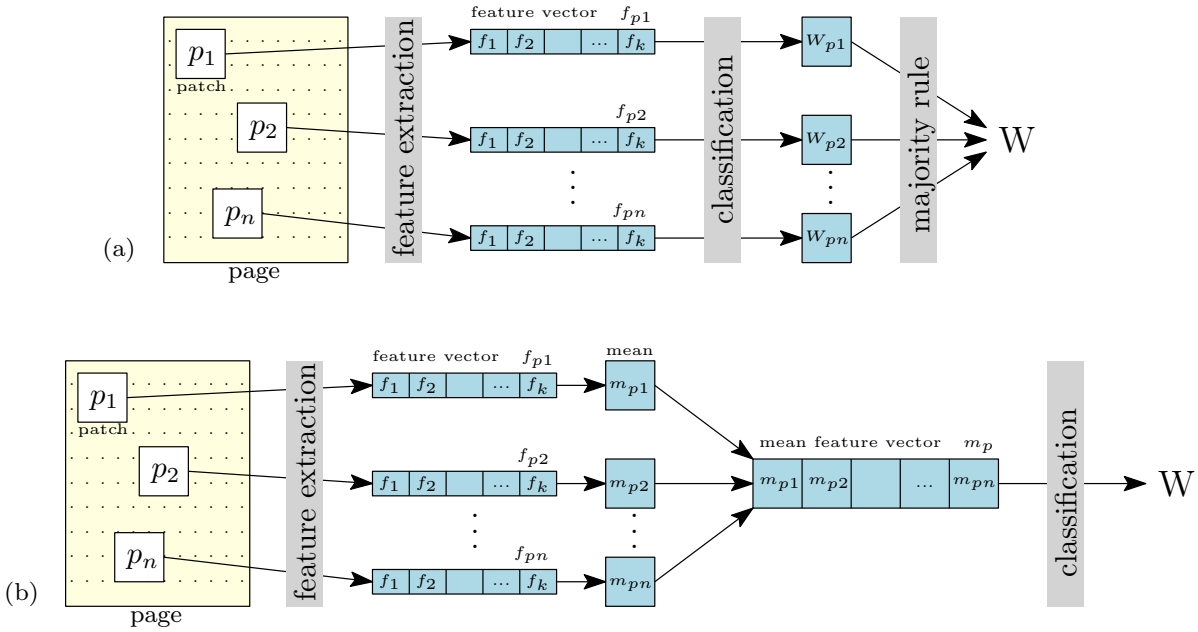


Figure 4.5: Writer identification strategies: (a) *Strategy-Major*, (b) *Strategy-Mean*.

Two types of patches $\text{patch}_{\text{char}}$ and $\text{patch}_{\text{allo}}$ (refer to *Section* 4.5) are used in both *Strategy-Major* and *Strategy-Mean* for writer identification.

4.7 Writer Verification

In this section, we discuss the writer verification task using handcrafted features followed by auto-derived features.

4.7.1 Handcrafted Feature-based Verification

In the writer verification task, we check whether two handwriting specimens are written by the same person or not. In fact, the goal is to find some distance measure between the two handwritten samples. If this distance is greater than a *decision threshold* T, then we infer that the samples are different, and the same, otherwise ($\leq T$) [122].

The distance measure is calculated using the handcrafted features generated from the handwritten sample. We used several distance measures such as *Minkowski* up to $order$ 5 (*Manhattan* when the $order = 1$, *Euclidean* when $order = 2$), *Bhattacharya*, *chi-square* (χ^2) and *Hausdorff* [39]. Here the chi-square (χ^2) distance worked well for our purpose.

The chi-square distance (χ^2_{ij}) between features obtained from two handwritten samples, i.e., *sample-i* and *sample-j*, are calculated as follows.

$$\chi^2_{ij} = \sum_{n=1}^{N} \frac{(f_{in} - f_{jn})^2}{(f_{in} + f_{jn})} \tag{4.6}$$

where, f_{in} and f_{jn} and are the feature vectors obtained from *sample-i* and *sample-j*, respectively. N denotes the dimension of the feature-vector and n represents the index.

In this writer verification task, two types of error are considered: *False Accept* (FA) and *False Reject* (FR). The nomenclatures of these errors depict their definition. FA is an error when two documents are falsely accepted as having the "same" source (written by the same writer), though actually they are "different". FR is the error when two documents are falsely rejected as "different" (written by different writers), when in fact they are written by the "same" person.

The error rates FAR (*False Acceptance Rate*) and FRR (*False Rejection Rate*) are calculated empirically by integration (up to/from the decision threshold T) of the distribution of distances between handwritten samples from different person $P_D(x)$ and the distribution of distances between samples of the same person $P_S(x)$, respectively [149].

$$FAR = \int_0^T P_D(x)dx \ , \quad FRR = \int_T^\infty P_S(x)dx \tag{4.7}$$

From FAR versus FRR plot, we obtained the EER (*Equal Error Rate*) where $FAR = FRR$. The writer verification performance in terms of *accuracy* or *balanced accuracy* is obtained as $(1 - EER) \times 100\%$.

4.7.2 Auto-derived Feature-based Verification

The Siamese Net [111] performs well for weakly supervised similarity metric learning and is successfully applied to various computer vision tasks, e.g., face verification [195], person re-identification [71], geo-localization [224], etc. Therefore, we use this net for writer verification using auto-derived features. Here, our task is treated as a binary classification to classify two handwritten specimens into "same" or "different" sourced.

Siamese Net contains identical twin neural architectures to produce two feature vectors from two images to be compared (Figure 4.6) [195, 224]. The neural networks of *Section* 4.5 are used for Siamese twins. Here, employing the *Strategy-Mean* of *Section* 4.6.2, we obtain the mean feature vector from a handwritten page H_i, considered as one subnet of the Siamese twins. Parallel to this, another mean feature vector is obtained by the other subnet of the Siamese twins from one more handwritten page H_j to be

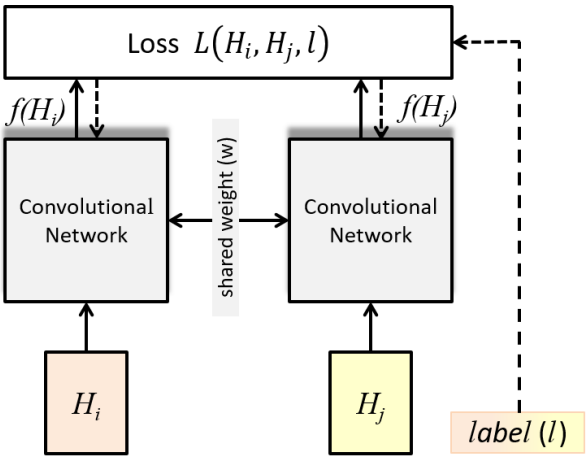


Figure 4.6: Siamese architecture.

compared with H_i. These twin architectures of the Siamese Net are joined by a *loss function* (L) at the top to train the similarity metric from the data. In this case, we use a margin-based loss function, i.e., *contrastive loss function* [180], which is given by:

$$L(H_i, H_j, l) = \alpha(1-l)D_w^2 + \beta l\{\max(0, m - D_w)\}^2 \tag{4.8}$$

where, *label* $l = 0$, if H_i and H_j are matched as the same, and $l = 1$, otherwise. Two constants α and β are chosen empirically as $\alpha = 0.5$ and $\beta = 0.5$. The margin $m > 0$ is set as the average squared pair distance. $D_w \equiv D_w(H_i, H_j) = \| f(H_i) - f(H_j) \|_2$ is the Euclidean distance between $f(H_i)$ and $f(H_j)$ which are two feature vectors generated by mapping of H_i and H_j to a real vector space through the convolutional network.

For performance evaluation, a threshold d is used on D_w to verify whether two handwritten samples are written by the "same" or a "different" writer. All the handwriting pairs (H_i, H_j), inferred to be written by the same writer are denoted as $\mathscr{P}_{same}$, whereas all pairs written by different writers are denoted as $\mathscr{P}_{diff}$.

Now, we define the set of true positives (TP) at d as follows.

$$TP(d) = \{(H_i, H_j) \in \mathscr{P}_{same},\ with\ D_w(H_i, H_j) \leq d\} \tag{4.9}$$

where all the handwriting pairs are correctly classified as the "same".

Similarly, when all the handwriting pairs are correctly classified as "different", then the set of true negatives (TN) at d is defined as below.

$$TN(d) = \{(H_i, H_j) \in \mathscr{P}_{diff},\ with\ D_w(H_i, H_j) > d\} \tag{4.10}$$

The true positive rate (TPR) and true negative rate (TNR) at d are computed as follows.

$$TPR(d) = \frac{|TP(d)|}{|\mathscr{P}_{same}|}, \quad TNR(d) = \frac{|TN(d)|}{|\mathscr{P}_{diff}|} \tag{4.11}$$

The overall accuracy or balanced accuracy is calculated as:

$$Accuracy = \max_{d \in D_w} \frac{TPR(d) + TNR(d)}{2}, \tag{4.12}$$

by varying d in the range of D_w with a step of 0.1.

For writer verification, we use page-level auto-derived features obtained from both patch types $\text{patch}_{\text{char}}$ and $\text{patch}_{\text{allo}}$, using *Strategy-Mean* of *Section* 4.6.2.

4.8 Experiments and Discussion

In this section, at first, we discuss the database employed and the data augmentation for its distribution among the training, validation and test sets. Then we present the results of writer identification and verification.

4.8.1 Database Employed

As mentioned in *Section* 4.2, we generated controlled (D_c) and uncontrolled (D_{uc}) databases, comprised of 600 pages each. D_c contained 3 sets (S_f, S_m, and S_s) of intra-variable writing. Each of these 3 sets contained 2 handwritten pages of 100 writers. Likewise, D_{uc} contained 3 sets S'_f, S'_m and S'_s, each having 2 pages by another set of 100 writers.

For intensive experimentation, we needed to augment our dataset. The data augmentation technique used here is presented below.

4.8.1.1 Data Augmentation

For augmenting our dataset, we were influenced by the idea of "*DropStroke*" [245] that was inspired by "Dropout" method from the deep neural network [89]. In [245], the *DropStroke* method was used to generate new data by omitting some strokes randomly from an *online* handwritten Chinese character.

Here, the *offline* data lacked the advantage of stroke drawing information of online data. Therefore, we remodeled the *DropStroke* as per our requirement for offline handwriting. We use the keypoint information here (refer to *Section* 4.4.3). The ink-pixel

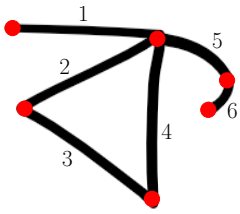


Figure 4.7: Pictorial representation of a Bengali character component: keypoints are marked by red dots and the six edges are numbered from '1' to '6'.

connection between two consecutive keypoints was considered as an edge/path/stroke. We dropped one edge from a text component (i.e., mostly character, obtained in *Section* 4.3) in such a strategy, so that the number of connected components did not increase. Thus, in Figure 4.7, edge '1' or '5' or '6' cannot be dropped since it will generate extra component; all the remaining edges can be dropped without any violation of this strategy.

To generate new samples from a page, we dropped $\lceil \alpha_d . n_d \rceil$ number of edges arbitrarily subject to the above condition. Here, n_d was the number of characters in a page and α_d was a parameter in a range of [0.1, 1], set empirically. The value of n_d was computed in *Section* 4.3.

Initially, a handwritten page was roughly horizontally split into two half-pages, which was a common technique for expansion of data samples in the writer identification task [248]. From each of these two half pages, we generated 10 different samples using our data augmentation technique. Therefore, a handwritten page produced 2 half-pages and 20 ($= 2 \times 10$) augmented handwritten samples, i.e., overall 22 ($= 2 + 20$) text samples (refer to Figure 4.8).

Thus, each of the databases D_c and D_{uc} contained 13200 (= 600 pages × 22 samples) text samples. Now, each of the subsets S_f, S_m, S_s and S'_f, S'_m, S'_s contained 44 (2 pages × 22 samples) text samples from each of the 100 writers.

We divided both databases D_c and D_{uc} into training, validation, and test sets with a 2:1:1 ratio. The set S_f was divided into S_{f1} (training), S_{f2} (validation) and S_{f3} (test) subsets. The S_{f1}, S_{f2}, S_{f3} contained 22, 11, 11 text samples, respectively, from each of the 100 writers. We ensured distinctiveness among training, validation and test sets, so that no common data was in between any pair of these sets. More elaborately, the S_{f1} contained 22 text samples generated from the full *page-1* of a writer in S_f. The subset S_{f2} contained 11 samples obtained from the top-half of the full *page-2*, and S_{f3} contained 11 samples generated from the bottom-half of this *page-2*. This is diagrammatically represented in Figure 4.8 for easy understanding. In this case, $S_f = S_{f1} \cup S_{f2} \cup S_{f3}$.

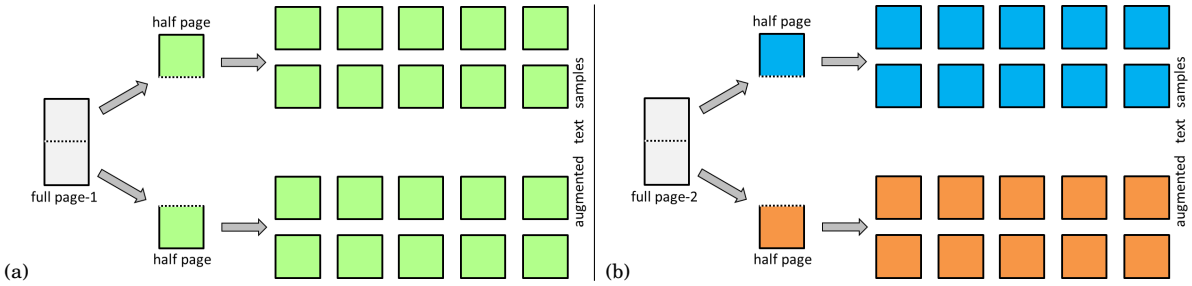


Figure 4.8: Augmented text samples, generated from 2 handwritten pages of a writer in set S_f. The subset S_{f1} contains green colored 22 samples for training, S_{f2} contains blue colored 11 samples for validation, and S_{f3} contains orange colored 11 samples for testing.

Similarly, S_m, S_s and S'_f, S'_m, S'_s sets were divided into training, validation and test sets. Here, $S_m = S_{m1} \cup S_{m2} \cup S_{m3}$, $S_s = S_{s1} \cup S_{s2} \cup S_{s3}$; $S'_f = S'_{f1} \cup S'_{f2} \cup S'_{f3}$, $S'_m = S'_{m1} \cup S'_{m2} \cup S'_{m3}$, $S'_s = S'_{s1} \cup S'_{s2} \cup S'_{s3}$.

Now, for our writer identification/verification task, we trained and tested with various types of intra-variable data. The experiments were performed in this way to imitate real-life situations, where a particular type of handwriting was absent.

4.8.2 Writer Identification Performance

In this subsection, we discuss the performance of the writer identification models based on handcrafted features and auto-derived features.

The writer identification accuracy was computed using a "Top-N" criterion, where the correct writer is marked at least one time within the 'N' ($\ll$ total number of writers) top-most classifier output confidences. Here, we computed results of Top-1, Top-2, and Top-5 criteria. However, in this chapter, Top-1 accuracies are presented in more detail for comparison among models. Top-2 and Top-5 accuracies are also presented when a better outcome is achieved.

4.8.2.1 Writer Identification Performance with Handcrafted Features

Here, we discuss the writer identification performance on both databases, D_c and D_{uc}, by employing handcrafted features.

4.8.2.1.1 Writer Identification by Handcrafted Features on D_c

The Top-1 writer identification performance on database D_c by employing feature F_{MM} with SVM classifier (say, system "$WI\_D_c\_F_{MM}$") is shown in Table 4.2.

Here, by training with all the training data ($S_{s1}+S_{m1}+S_{f1}$) of D_c, and testing only on S_{s3}, we obtained a 62.78% accuracy. Likewise, training on $S_{s1}+S_{m1}+S_{f1}$, and testing on S_{m3} and S_{f3}, we obtained 62.23% and 61.31% accuracies, respectively.

Experimenting on the same type set yielded better accuracy, e.g., training on subset S_{s1} and testing on subset S_{s3} (say, experimental setup S_{s1}/S_{s3} or, E_{ss}) provided 61.23% accuracy, where both training and testing are parts of the set S_s. Similarly, experimental setup S_{m1}/S_{m3} (E_{mm}) and S_{f1}/S_{f3} (E_{ff}) provides a 60.48% and 59.32% accuracy, respectively.

Next, we looked at the performance of experiments on different training and test sets. For example, training on S_{s1} and testing on S_{m3} (say, experimental setup S_{s1}/S_{m3} or, E_{sm}) yielded 37.17% accuracy, which was quite low. The reverse experimental setup, i.e., S_{m1}/S_{s3} (E_{ms}) also showed poor performance (38.45% accuracy). Similarly, experimental setups E_{mf}, E_{fm}, E_{sf}, E_{fs} provided poor results, with accuracies 36.67%, 34.59%, 29.49%, 31.45%, respectively. The reason behind such a low outcome was the presence of high variability between the training and test sets. Experimental setups E_{sf} and E_{fs} showed the lowest performance, since they contained highly intra-variable writing.

For our task, we defined the performance of our model by a tuple of 9 major accuracies (%) obtained by various experimental setups. This 9-tuple was (AE_{ss}, AE_{mm}, AE_{ff}, AE_{smv}, AE_{sfv}, AE_{mfv}, $AE_{smf/s}$, $AE_{smf/m}$, $AE_{smf/f}$) which was used to compare multiple models used in this chapter. AE_{ss}, AE_{mm}, AE_{ff} were the accuracy measures obtained from the E_{ss}, E_{mm}, E_{ff} experimental setups, respectively. These AE_{ss}, AE_{mm}, AE_{ff} accuracies showed the efficacy of a model on low intra-variable handwritings which

Table 4.2: Top-1 writer identification performance of system $WI\_D_c\_F_{MM}$

Set	***Accuracy (%)***		
Training \ ***Test***	S_{s3}	S_{m3}	S_{f3}
S_{s1}	61.23	37.17	29.49
S_{m1}	38.45	60.48	36.67
S_{f1}	31.45	34.59	59.32
$S_{s1}+S_{m1}$	61.42	61.89	37.04
$S_{s1}+S_{f1}$	62.24	38.47	60.33
$S_{m1}+S_{f1}$	40.72	60.95	59.86
$S_{s1}+S_{m1}+S_{f1}$	**62.78**	62.23	61.31

Table 4.3: Top-1 writer identification performance of system $WI\_D_c\_F_{DH}$

Set	***Accuracy (%)***		
Training \ ***Test***	S_{s3}	S_{m3}	S_{f3}
S_{s1}	72.67	44.71	37.29
S_{m1}	45.53	71.86	44.17
S_{f1}	39.23	43.59	70.93
$S_{s1}+S_{m1}$	72.92	72.84	44.85
$S_{s1}+S_{f1}$	72.86	45.42	71.54
$S_{m1}+S_{f1}$	46.28	72.37	71.72
$S_{s1}+S_{m1}+S_{f1}$	73.49	73.36	72.26

were mostly similar types. In Table 4.2, we show these accuracies highlighted in green, which can be seen better in the softcopy of this thesis.

AE_{smv} was the average (arithmetic mean) accuracy obtained from experimental setups E_{sm} (S_{s1}/S_{m3}) and E_{ms} (S_{m1}/S_{s3}). This was depicted with red shade in Table 4.2. Similarly, AE_{sfv} was obtained from E_{sf}, E_{fs} (blue shaded in Table 4.2) and AE_{mfv} was obtained from E_{mf}, E_{fm} (yellow in Table 4.2), respectively. AE_{smv}, AE_{sfv}, AE_{mfv} demonstrated the system performance when both training and test sets contained highly intra-variable writing.

$AE_{smf/s}$ was the accuracy obtained from the experimental setup $E_{smf/s}$, where training was performed on $S_{s1}+S_{m1}+S_{f1}$, and testing was executed on S_{s3}. $AE_{smf/m}$ and $AE_{smf/f}$ were obtained by testing on S_{m3} and S_{f3}, respectively, while training was performed on $S_{s1}+S_{m1}+S_{f1}$, similar to the training of $AE_{smf/s}$. Here, $AE_{smf/s}$, $AE_{smf/m}$, $AE_{smf/f}$ showed the performance when the system was trained with all available handwriting varieties of an individual. In Table 4.2, we show these accuracies in gray shade.

The performance of the system $WI\_D_c\_F_{MM}$ in terms of 9-tuple is (61.23, 60.48, 59.32, 37.81, 30.47, 35.63, 62.78, 62.23, 61.31).

The Top-1 writer identification performance on database D_c using feature F_{DH} with SVM (say, system "$WI\_D_c\_F_{DH}$") is shown in Table 4.3. The performance of system $WI\_D_c\_F_{DH}$ in terms of 9-tuple is (72.67, 71.86, 70.93, 45.12, 38.26, 43.88, 73.49, 73.36, 72.26). This can be tallied with Table 4.3, as we tallied $WI\_D_c\_F_{MM}$ performance with Table 4.2.

The Top-1 writer identification performance on database D_c using feature F_{DC} with SVM (say, system "$WI\_D_c\_F_{DC}$") is shown in Table 4.4. The performance of system $WI\_D_c\_F_{DC}$ in terms of 9-tuple is (71.54, 70.25, 69.71, 43.56, 36.38, 40.80, 72.48, 71.85, 70.93). This can be tallied with Table 4.4.

Combining Tables 4.2, 4.3 and 4.4, we generate Table 4.5 to present only the 9-tuple accuracies of all handcrafted feature-based writer identification models dealing with D_c,

Table 4.4: Top-1 writer identification performance of system $WI\_D_c\_F_{DC}$

Set	***Accuracy (%)***		
Test / ***Training***	S_{s3}	S_{m3}	S_{f3}
S_{s1}	71.54	43.20	37.04
S_{m1}	43.92	70.25	41.06
S_{f1}	35.72	40.54	69.71
$S_{s1}+S_{m1}$	71.75	70.84	41.74
$S_{s1}+S_{f1}$	71.66	43.37	70.14
$S_{m1}+S_{f1}$	44.47	71.45	70.68
$S_{s1}+S_{m1}+S_{f1}$	72.48	71.85	70.93

Table 4.5: Top-1 writer identification performance using handcrafted features on D_c

Model	***9-tuple Accuracy (%)***									***Rank***
	AE_{ss}	AE_{mm}	AE_{ff}	AE_{smv}	AE_{sfv}	AE_{mfv}	$AE_{smf/s}$	$AE_{smf/m}$	$AE_{smf/f}$	
$WI\_D_c\_F_{MM}$	61.23	60.48	59.32	37.81	30.47	35.63	62.78	62.23	61.31	3
$WI\_D_c\_F_{DH}$	72.67	71.86	70.93	45.12	38.26	43.88	73.49	73.36	72.26	1
$WI\_D_c\_F_{DC}$	71.54	70.25	69.71	43.56	36.38	40.80	72.48	71.85	70.93	2

for comparison and easy visualization.

The models were ranked using the *Borda count* [170]. Here, the models were initially ranked with respect to each accuracy of the 9-tuple (i.e., performance on each experimental setup), then the aggregate ranking was computed by the *max* rule. If there was a draw between two models, then we provided weightage on the accuracies AE_{smv}, AE_{sfv}, AE_{mfv}. The aggregate ranks are shown in the last columns of Tables 4.5 - 4.8. Here, Rank '1' denotes first, i.e., the best performing model, Rank '2' indicates the second best performing model, and so on.

Although we computed 21 accuracy measures as in Tables 4.2 - 4.4, we present the 9-tuple accuracy measures like Table 4.5 further for model assessment and for simple visualization.

4.8.2.1.2 Writer Identification by Handcrafted Features on D_{uc}

By employing database D_{uc}, here also, we generated three similar handcrafted feature-based writer identification models as mentioned in *Section* 4.8.2.1.1. In Table 4.6, we present the performance of these models on D_{uc} with respect to the 9-tuple accuracy.

In this case, while employing D_{uc}, the AE_{ss} accuracy was obtained from an experimental setup E_{ss} where S'_{s1} was used for training and S'_{s3} was used for testing. Similarly, for obtaining $AE_{smf/m}$, training was performed on $(S'_{s1} + S'_{m1} + S'_{f1})$, and testing was executed on S'_{m3}. Likewise, other accuracies of 9-tuple were obtained (refer to *Section* 4.8.2.1.1).

Table 4.6: Top-1 writer identification performance using handcrafted features on D_{uc}

Model	***9-tuple Accuracy (%)***									***Rank***
	AE_{ss}	AE_{mm}	AE_{ff}	AE_{smv}	AE_{sfv}	AE_{mfv}	$AE_{smf/s}$	$AE_{smf/m}$	$AE_{smf/f}$	
$WI\_D_{uc}\_F_{MM}$	60.08	58.51	57.72	37.01	29.77	33.92	61.63	61.04	60.01	3
$WI\_D_{uc}\_F_{DH}$	71.79	71.21	70.19	43.85	37.01	42.21	72.93	72.66	71.97	1
$WI\_D_{uc}\_F_{DC}$	69.77	68.69	68.54	42.43	34.96	39.16	70.82	70.02	69.66	2

Experimenting on both the databases D_c and D_{uc} using handcrafted features, overall the F_{DH} feature-based model performed the best and the F_{MM} feature-based model achieved the lowest results.

It can be observed from Tables 4.5 and 4.6 that the accuracies AE_{smv}, AE_{sfv}, AE_{mfv} are very low. We noted on AE_{smv}, AE_{sfv}, AE_{mfv} that Top-2 (Top-5) writer identification performance provided additional at most 0.38% (2.97%) and 0.30% (2.53%) accuracy on D_c and D_{uc}, respectively.

4.8.2.2 Writer Identification Performance with Auto-derived Features

We perform writer identification by feeding $\text{patch}_{\text{char}}$ to the Basic_CNN with the *Strategy-Major*, and call this model: "BCNN_char_major". Likewise, feeding $\text{patch}_{\text{char}}$ to the Basic_CNN with the *Strategy-Mean*, is called a "BCNN_char_mean" model. The $\text{patch}_{\text{allo}}$s when input into the Basic_CNN with the *Strategy-Major* and the *Strategy-Mean*, are called the "BCNN_allo_major" and the "BCNN_allo_mean", respectively.

Thus, a convolutional network produces 4 variations of auto-derived feature-based models for writer identification, i.e., "x_char_major", "x_char_mean", "x_allo_major", and "x_allo_mean". Here, 'x' is to be replaced by the convolutional network name. The 'x' is replaced by 'SN', 'GN', 'XN', 'VN', 'RN' while employing SqueezeNet, GoogLeNet, Xception Net, VGG-16, ResNet-101, respectively. For example, feeding $\text{patch}_{\text{char}}$ in GoogLeNet with the *Strategy-Mean* is called: "GN_char_mean".

Consequently, 6 types of convolutional networks, each of 4 various configurations, produce a total of 24 ($= 6 \times 4$) models. Here, we present the previously mentioned 9-tuple accuracy measure for each model (refer to *Section* 4.8.2.1.1).

4.8.2.2.1 Writer Identification by Auto-derived Features on D_c

In Table 4.7, we present the 9-tuple writer identification accuracies of auto-derived feature-based models performing on the D_c database.

The ranks of the models are shown in the rightmost column of Table 4.7. Here, the XN_allo_mean model performed the best for writer identification on the D_c database. Although in this model, the AE_{ss}, $AE_{smf/s}$ accuracies were more than 97%, the performance was comparatively lower with respect to AE_{smv}, AE_{sfv}, AE_{mfv} accuracies.

According to Table 4.7, the overall *Strategy-Mean* worked better than *Strategy-Major*. In general, the $\text{patch}_{\text{allo}}$ with the *Strategy-Mean* worked better, but $\text{patch}_{\text{allo}}$ with *Strategy-Major* did not work so well.

Table 4.7: Top-1 writer identification performance using auto-derived features on D_c

	Model (WI_D_c)	***9-tuple Accuracy (%)***									***Rank***
		AE_{ss}	AE_{mm}	AE_{ff}	AE_{smv}	AE_{sfv}	AE_{mfv}	$AE_{smf/s}$	$AE_{smf/m}$	$AE_{smf/f}$	
Basic CNN	BCNN_char_major	81.25	81.72	81.51	57.58	50.03	53.67	83.72	83.32	82.58	23
	BCNN_allo_major	81.78	81.48	81.09	58.25	48.92	53.35	84.16	82.35	81.89	24
	BCNN_char_mean	83.56	83.12	82.37	60.02	51.27	55.93	85.58	84.73	83.89	22
	BCNN_allo_mean	84.53	83.63	82.92	60.59	51.34	55.25	86.21	85.33	84.74	21
Squeeze Net	SN_char_major	87.27	85.54	85.14	62.92	53.15	58.03	88.16	87.69	86.29	19
	SN_allo_major	86.75	84.55	85.37	62.25	52.71	57.35	87.25	85.56	87.02	20
	SN_char_mean	88.46	87.79	87.31	65.35	56.79	61.53	90.27	89.35	89.05	18
	SN_allo_mean	89.23	88.48	87.72	66.56	56.05	61.45	90.53	89.72	89.25	17
GoogLe Net	GN_char_major	91.09	90.07	87.97	67.37	57.08	62.58	91.60	91.44	90.44	13
	GN_allo_major	90.29	90.16	87.48	66.28	56.68	61.42	90.81	90.79	90.07	16
	GN_char_mean	91.34	89.78	89.29	68.04	58.28	63.99	92.58	90.62	91.20	12
	GN_allo_mean	91.39	90.48	90.04	69.12	58.92	63.24	93.13	92.16	91.25	11
VGG -16	VN_char_major	91.02	89.25	88.58	67.33	57.53	62.22	91.71	90.03	90.43	**15**
	VN_allo_major	90.62	89.57	88.48	66.76	57.80	62.26	91.49	90.73	90.51	**14**
	VN_char_mean	91.91	91.24	90.47	68.27	59.13	64.17	92.53	92.07	92.29	10
	VN_allo_mean	92.74	90.91	91.29	69.23	60.04	64.37	93.43	91.89	92.63	9
ResNet -101	RN_char_major	92.01	92.43	91.76	69.27	59.66	65.24	93.62	92.97	92.86	7
	RN_allo_major	92.44	91.72	91.05	69.69	59.61	64.98	93.42	92.33	92.27	8
	RN_char_mean	93.26	93.49	92.83	70.99	61.34	66.21	94.51	93.77	94.42	6
	RN_allo_mean	94.07	93.66	92.93	71.32	61.63	66.42	95.25	94.72	94.34	5
Xception Net	XN_char_major	95.56	94.53	93.73	71.47	62.51	67.54	95.51	95.73	94.82	3
	XN_allo_major	94.83	93.66	93.50	70.93	62.58	66.83	95.07	95.19	94.48	4
	XN_char_mean	96.73	95.46	95.12	72.75	63.74	68.66	97.04	96.97	96.12	2
	XN_allo_mean	97.02	95.71	95.47	73.74	64.12	68.94	97.87	96.46	96.84	1

4.8.2.2.2 Writer Identification by Auto-derived Features on D_{uc}

In Table 4.8, we present the 9-tuple writer identification accuracies of auto-derived feature-based models performing on the D_{uc} database.

Here also, from Table 4.8, we noted that performance on the intra-variable handwritten sample with different training and testing sample types was not so well, i.e., AE_{smv}, AE_{sfv}, AE_{mfv} accuracies were comparatively low. The model XN_allo_mean produced the best outcome. The other model rankings can be observed in the rightmost column of Table 4.8. In this case, it can be noted that the overall *Strategy-Mean* worked better than the *Strategy-Major*.

Comparing Tables 4.7 and 4.8, it can be observed that the general performance on database D_c is better than D_{uc}. The rank orders are almost similar in cases of Tables 4.7 and 4.8. Here, only the ranks of model VN_char_major and VN_allo_major are interchanged. This scenario suggests that our model is quite stable for different databases.

All the auto-derived feature-based models worked better than handcrafted feature-based models. However, the AE_{smv}, AE_{sfv}, AE_{mfv} accuracies were also low here in comparison with the other accuracies of the 9-tuple. Using auto-derived features, the Top-2 (Top-5) writer identification accuracies of AE_{smv}, AE_{sfv}, AE_{mfv} increased at most

Table 4.8: Top-1 writer identification performance using auto-derived features on D_{uc}

	Model (WI_D_{uc})	***9-tuple Accuracy (%)***									***Rank***
		AE_{ss}	AE_{mm}	AE_{ff}	AE_{smv}	AE_{sfv}	AE_{mfv}	$AE_{smf/s}$	$AE_{smf/m}$	$AE_{smf/f}$	
Basic CNN	BCNN_char_major	80.69	80.53	80.43	56.26	47.65	52.59	82.19	80.77	80.72	23
	BCNN_allo_major	79.57	79.94	79.31	57.10	48.98	52.33	82.65	81.38	80.49	24
	BCNN_char_mean	81.96	82.05	82.78	58.16	49.47	53.04	84.22	83.23	82.01	22
	BCNN_allo_mean	83.01	81.73	81.64	59.5	50.02	54.15	85.13	84.24	82.81	21
Squeeze Net	SN_char_major	86.06	83.91	83.93	60.88	51.78	56.81	86.47	86.13	85.11	19
	SN_allo_major	85.25	83.51	83.29	62.26	50.96	55.71	86.13	85.43	83.85	20
	SN_char_mean	88.12	86.82	86.78	63.80	54.45	58.53	89.48	89.25	88.98	18
	SN_allo_mean	89.33	88.03	87.06	65.27	54.84	60.17	90.57	89.54	88.57	17
GoogLe Net	GN_char_major	90.31	89.96	87.82	66.14	55.75	60.79	90.85	91.34	89.82	13
	GN_allo_major	89.77	88.83	86.96	64.45	55.73	61.51	90.15	89.93	89.09	16
	GN_char_mean	90.43	90.47	88.39	66.41	56.48	60.86	92.14	90.79	91.07	12
	GN_allo_mean	90.54	89.46	89.24	67.39	57.59	61.94	91.82	91.41	90.45	11
VGG -16	VN_char_major	90.37	89.55	87.51	65.67	55.82	60.54	91.49	90.69	90.52	**14**
	VN_allo_major	89.74	88.85	88.63	66.92	55.64	60.41	91.61	89.20	90.41	**15**
	VN_char_mean	91.60	90.18	90.38	67.09	57.39	63.12	90.78	90.49	91.65	10
	VN_allo_mean	92.69	90.34	90.55	67.94	58.46	63.02	92.52	91.55	91.46	9
ResNet -101	RN_char_major	91.82	92.72	90.28	68.16	58.95	63.42	93.16	92.96	92.35	7
	RN_allo_major	91.85	91.42	90.73	68.60	57.41	63.07	92.42	91.41	91.97	8
	RN_char_mean	92.72	93.38	92.27	69.03	60.02	64.64	93.79	93.01	93.57	6
	RN_allo_mean	94.13	93.42	92.36	69.45	59.63	65.13	94.47	94.21	93.48	5
Xception Net	XN_char_major	95.13	94.31	92.82	69.15	60.89	65.77	95.27	95.34	93.85	3
	XN_allo_major	94.62	93.01	92.51	69.95	61.51	65.78	94.42	94.84	94.16	4
	XN_char_mean	96.49	94.56	94.56	70.85	61.85	67.81	96.25	96.04	95.74	2
	XN_allo_mean	96.81	95.52	95.03	72.52	62.79	66.53	97.09	96.59	95.62	1

1.56% (7.76%) and 1.35% (6.89%) for D_c and D_{uc}, respectively.

In Figure 4.9 - Figure 4.10, we present the radar plots of 9-tuple accuracies of writer identification shown in Tables 4.5 - 4.8.

4.8.3 Writer Verification Performance

In this section, we discuss writer verification performances of the models employed using both handcrafted features and auto-derived features.

The writer verification accuracies were obtained as mentioned in *Section* 4.7. Similar to writer identification, here we also used multiple experimental setups and finally obtained 9-tuple accuracies (refer to *Section* 4.8.2.1). The *Borda count* [170] was also used here to rank the models as shown in the last columns of Tables 4.9 - 4.12.

4.8.3.1 Writer Verification Performance with Handcrafted Features

Similar to the three writer identification models employing handcrafted features (refer to *Section* 4.8.2.1), here also we obtained three writer verification models. These writer verification models were evaluated on both the databases D_c and D_{uc}. The procedure of model accuracy computation is described in *Section* 4.7.1. The performance of the models is discussed as follows.

4.8.3.1.1 Writer Verification by Handcrafted Features on D_c

We obtained $WV\_D_c\_F_{MM}$, $WV\_D_c\_F_{DH}$, $WV\_D_c\_F_{DC}$ models for writer verification similar to the writer identification models (refer to *Section* 4.8.2.1.1). For example, $WV\_D_c\_F_{MM}$ was such a writer verification model, where F_{MM} handcrafted features were used on database D_c.

In Table 4.9, we present the writer verification performance of these three models in terms of 9-tuple accuracy. The ranks of these models are mentioned in the rightmost column of Table 4.9. Here, the model $WV\_D_c\_F_{DH}$ performed the best.

Table 4.9: Writer verification performance using handcrafted features on D_c

Model	***9-tuple Accuracy (%)***									***Rank***
	AE_{ss}	AE_{mm}	AE_{ff}	AE_{smv}	AE_{sfv}	AE_{mfv}	$AE_{smf/s}$	$AE_{smf/m}$	$AE_{smf/f}$	
$WV\_D_c\_F_{MM}$	72.07	71.15	69.92	48.50	36.58	41.43	73.66	73.12	71.34	3
$WV\_D_c\_F_{DH}$	86.64	86.06	85.69	53.72	43.57	48.18	87.87	87.57	86.02	1
$WV\_D_c\_F_{DC}$	83.60	82.85	82.48	52.46	41.09	46.08	84.85	84.21	82.80	2

4.8.3.1.2 Writer Verification by Handcrafted Features on D_{uc}

Here also, we have generated three handcrafted feature-based writer verification models $WV\_D_{uc}\_F_{MM}$, $WV\_D_{uc}\_F_{DH}$, $WV\_D_{uc}\_F_{DC}$ for experimentation on database D_{uc}.

In Table 4.10, the writer verification results of these three models are presented, where the rightmost column shows their ranking. In this case, the model $WV\_D_{uc}\_F_{DH}$ performed the best.

Table 4.10: Writer verification performance using handcrafted features on D_{uc}

Model	***9-tuple Accuracy (%)***									***Rank***
	AE_{ss}	AE_{mm}	AE_{ff}	AE_{smv}	AE_{sfv}	AE_{mfv}	$AE_{smf/s}$	$AE_{smf/m}$	$AE_{smf/f}$	
$WV\_D_{uc}\_F_{MM}$	69.58	69.05	67.09	47.96	35.76	40.26	71.43	70.12	69.34	3
$WV\_D_{uc}\_F_{DH}$	84.87	84.45	84.52	52.34	40.88	46.50	86.20	86.24	84.03	1
$WV\_D_{uc}\_F_{DC}$	81.50	80.38	81.12	50.86	39.32	45.74	83.09	83.05	81.34	2

For experimentation on both databases D_c and D_{uc}, the rankings are similar when using the same feature-based model. Overall, the F_{DH} feature-based model performed the best and the F_{MM} feature-based model achieved the lowest results for verification also.

It can be observed from Tables 4.9 and 4.10 that the accuracies AE_{smv}, AE_{sfv}, AE_{mfv} are very low in comparison with other accuracies of 9-tuple. Here, the ranks in Tables

4.9 and 4.10 are the same for similar handcrafted feature-based models employed in D_c and D_{uc}.

4.8.3.2 Writer Verification Performance with Auto-derived Features

In this section, we discuss the performance of auto-derived feature-based writer verification models. The procedure to obtain the verification accuracy using auto-derived features is mentioned in *Section* 4.7.2. Here, only the *Strategy-Mean* was used, since the *Strategy-Major* performed insignificantly. Therefore, we obtained 12 models from 6 types of convolutional networks, with 2 forms of patch ($\text{patch}_{\text{char}}$ and $\text{patch}_{\text{allo}}$), fed using only one strategy. The naming convention of these verification models was kept similar to the writer identification models.

4.8.3.2.1 Writer Verification by Auto-derived Features on D_c

In Table 4.11, we present the 9-tuple writer verification accuracies using auto-derived features while experimenting on D_c. Here, XN_allo_mean performed the best. All the model rankings are presented in the rightmost column of Table 4.11. In general, $\text{patch}_{\text{allo}}$ worked better than $\text{patch}_{\text{char}}$. Only for VGG-16, the $\text{patch}_{\text{char}}$ performed better than $\text{patch}_{\text{allo}}$.

Table 4.11: Writer verification performance using auto-derived features on D_c

Model (WV_D_c)	***9-tuple Accuracy (%)***									***Rank***
	AE_{ss}	AE_{mm}	AE_{ff}	AE_{smv}	AE_{sfv}	AE_{mfv}	$AE_{smf/s}$	$AE_{smf/m}$	$AE_{smf/f}$	
BCNN_char_mean	92.22	91.48	92.13	69.19	60.94	66.59	94.17	93.53	93.44	12
BCNN_allo_mean	92.45	91.78	92.75	70.69	61.84	66.28	93.33	92.12	92.72	11
SN_char_mean	95.74	94.85	95.53	74.85	65.14	70.97	97.24	97.31	96.55	10
SN_allo_mean	96.09	95.09	96.52	77.08	66.16	70.75	96.53	95.93	96.07	9
GN_char_mean	96.55	95.60	96.21	75.39	65.97	71.42	97.97	97.97	96.73	**8**
GN_allo_mean	96.24	95.83	96.55	77.90	66.18	71.47	97.38	96.79	96.58	**7**
VN_char_mean	97.77	96.36	97.02	75.80	66.69	72.19	98.39	98.17	97.67	**5**
VN_allo_mean	97.20	96.81	96.73	78.04	66.48	72.35	97.82	96.96	97.08	**6**
RN_char_mean	98.21	97.85	98.24	77.48	67.83	73.28	99.50	99.47	99.05	4
RN_allo_mean	98.74	98.15	98.27	79.72	68.36	73.51	99.76	98.73	98.95	3
XN_char_mean	98.95	98.79	98.82	78.68	69.70	75.24	99.76	99.68	99.27	2
XN_allo_mean	99.24	99.06	98.68	80.79	70.02	74.98	99.84	99.73	99.18	1

4.8.3.2.2 Writer Verification by Auto-derived Features on D_{uc}

On database D_{uc}, the 9-tuple writer verification accuracies of various auto-derived feature-based models are shown in Table 4.12. In this case, XN_allo_mean performed

the best. The ranks of other models are shown in the rightmost column of Table 4.12. Overall, $\text{patch}_{\text{allo}}$ worked better than $\text{patch}_{\text{char}}$ except for VGG-16 and GoogLeNet.

Table 4.12: Writer verification performance using auto-derived features on D_{uc}

Model (WV_D_{uc})	***9-tuple Accuracy (%)***									***Rank***
	AE_{ss}	AE_{mm}	AE_{ff}	AE_{smv}	AE_{sfv}	AE_{mfv}	$AE_{smf/s}$	$AE_{smf/m}$	$AE_{smf/f}$	
BCNN_char_mean	91.55	90.57	91.26	69.15	60.17	65.54	93.24	92.57	92.06	12
BCNN_allo_mean	91.78	90.92	91.78	70.29	61.31	65.91	92.63	91.81	92.53	11
SN_char_mean	95.36	94.25	94.54	74.77	64.15	69.98	96.50	96.95	96.12	10
SN_allo_mean	95.86	94.46	96.50	76.84	65.35	70.54	96.06	95.90	95.47	9
GN_char_mean	95.86	95.29	96.02	74.95	65.36	71.27	97.07	97.09	96.67	**7**
GN_allo_mean	95.55	95.29	96.54	77.14	65.81	70.52	97.23	96.76	96.12	**8**
VN_char_mean	97.36	96.02	96.72	75.10	66.58	71.62	97.06	97.94	97.62	**5**
VN_allo_mean	96.43	96.75	95.99	77.28	66.18	72.34	97.67	96.61	96.45	**6**
RN_char_mean	97.46	97.63	98.02	77.15	67.22	73.03	99.30	99.29	98.53	4
RN_allo_mean	98.44	98.03	98.27	79.40	68.26	73.37	98.89	97.79	98.13	3
XN_char_mean	98.73	98.69	98.06	78.13	69.29	74.28	99.65	98.96	98.77	2
XN_allo_mean	98.55	98.73	97.89	79.84	69.80	74.76	99.19	99.54	98.54	1

From Tables 4.11 and 4.12, it can be observed that the AE_{smv}, AE_{sfv}, AE_{mfv} accuracies are very low in comparison with other accuracies of 9-tuple. Here, Tables 4.11 and 4.12 depict similar ranks except for the GoogLeNet-based models.

In Figure 4.11 - Figure 4.12, we present the radar plots of 9-tuple accuracies of writer verification shown in Tables 4.9 - 4.12.

4.8.4 Observations

From the above experiments (refer to *Section*s 4.8.2 and 4.8.3, Tables 4.5 - 4.12, Figure 4.9 - Figure 4.12), our major observations are noted as follows.

(i) Among the accuracies in 9-tuple, the AE_{smv}, AE_{sfv}, AE_{mfv} accuracies were comparatively low for all the models used for *writer identification*/*verification*. It suggests that if the training and test set contain similar types of samples, the models can perform better.

Moreover, here the AE_{sfv} accuracy was lower than AE_{smv} and AE_{mfv}. It indicates that sets S_s and S_f of D_c (S'_s and S'_f sets of D_{uc}) contain higher intra-variable writing than the other set combinations.

(ii) The auto-derived features worked better than the handcrafted features for the *writer identification*/*verification*. The reason may be the high dimensionality of the auto-derived features and the use of deep convolutional architectures.

Among the handcrafted features, F_{DH} performed the best, while F_{DC} performed better than F_{MM}. Among auto-derived features, Xception Net-based features worked the best.

(iii) For auto-derived feature-based *writer identification* models, mostly the *Strategy-Mean* worked better than *Strategy-Major*.

(iv) In general, for auto-derived feature-based *writer identification* models, $\text{patch}_{\text{allo}}$ with *Strategy-Mean* worked the best, whereas $\text{patch}_{\text{allo}}$ with *Strategy-Major* performed lowest. Combining $\text{patch}_{\text{char}}$, *Strategy-Mean* worked better than *Strategy-Major*. In other words, using combinations of *patch* and *Strategy*, the overall performance in highest to lowest order is as follows: allo_mean $\succ$ char_mean $\succ$ char_major $\succ$ allo_major.

(v) For auto-derived feature-based *writer verification* models, mostly the $\text{patch}_{\text{allo}}$ worked better than the $\text{patch}_{\text{char}}$.

(vi) Altogether, the *writer identification / verification* performance on the controlled database (D_c) was better than the uncontrolled database (D_{uc}).

(vii) As a whole, all the *writer identification / verification* models provided quite similar AE_{ss}, AE_{mm}, AE_{ff} accuracies (differences range up to 3.12% Top-1). This implies that our system is quite robust in working with various handwriting types.

(viii) In general, the handcrafted feature-based models and auto-derived feature-based models for *writer identification / verification* followed the same trend, respectively (refer to Tables 4.5 - 4.12). It can be visualized by the radar plots of Figure 4.9 - Figure 4.12. Here, the individual radar plot follows almost the same trend with creating a band of a certain width.

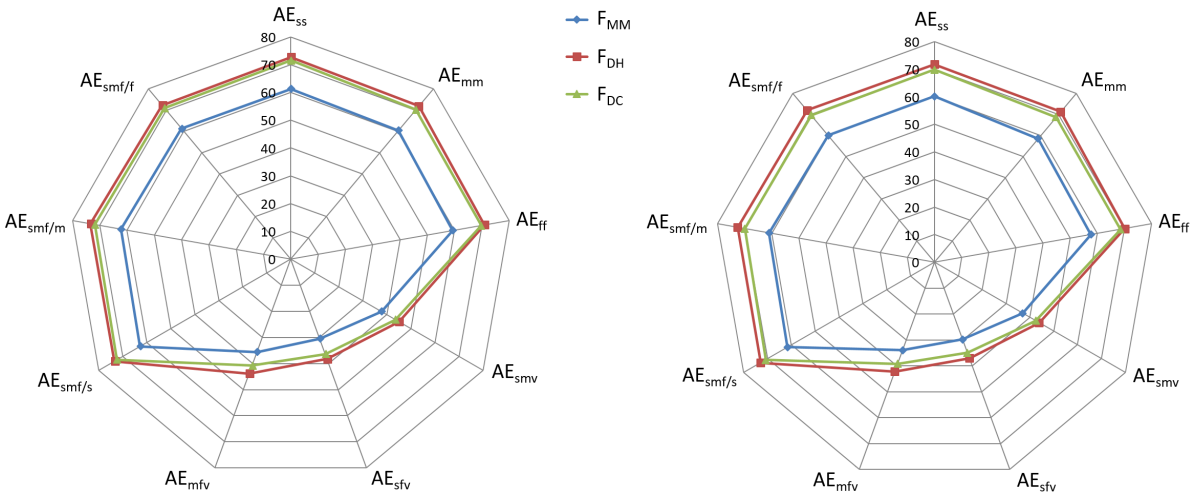


Figure 4.9: Radar plot of Top-1 writer identification performance using handcrafted features. *Left*: representing Table 4.5 on D_c and *Right*: representing Table 4.6 on D_{uc}.

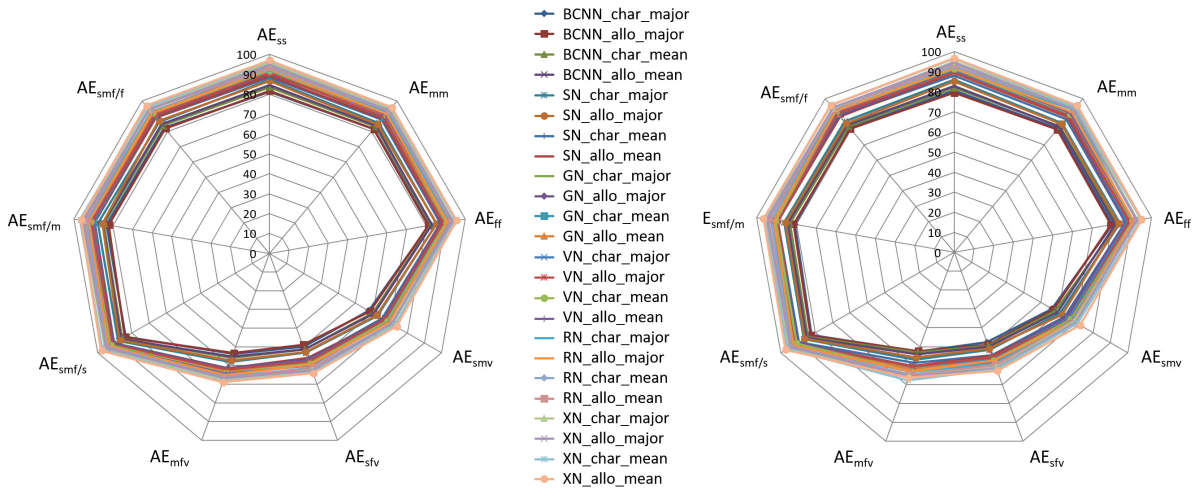


Figure 4.10: Radar plot of Top-1 writer identification performance using auto-derived features. *Left*: representing Table 4.7 on D_c and *Right*: representing Table 4.8 on D_{uc}.

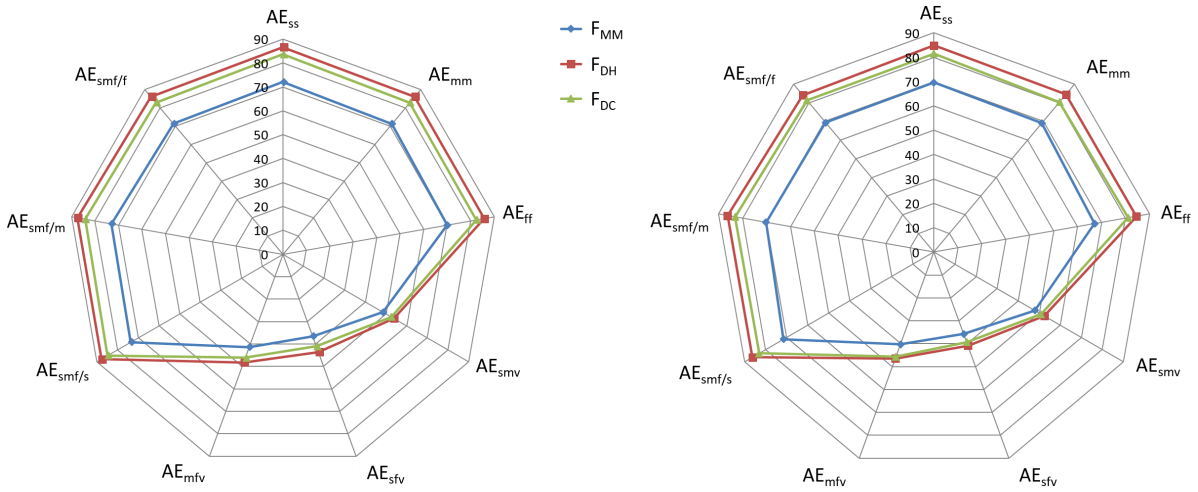


Figure 4.11: Radar plot of writer verification performance using handcrafted features. *Left*: representing Table 4.9 on D_c and *Right*: representing Table 4.10 on D_{uc}.

4.8.5 Writer Identification/Verification by Pre-training

From the previous experiments, we observed that AE_{smv}, AE_{sfv}, AE_{mfv} accuracies were relatively lower than other accuracies of the 9-tuple. Therefore, in this section, we aim to increase these three accuracies (AE_{smv}, AE_{sfv}, AE_{mfv}), say, by 3-tuple.

We noted that the auto-derived features worked better than the handcrafted features. Therefore, here we focus only on auto-derived features. We also observed that the *Xception Net* performed the best among all models for our problem. Hence, we continue

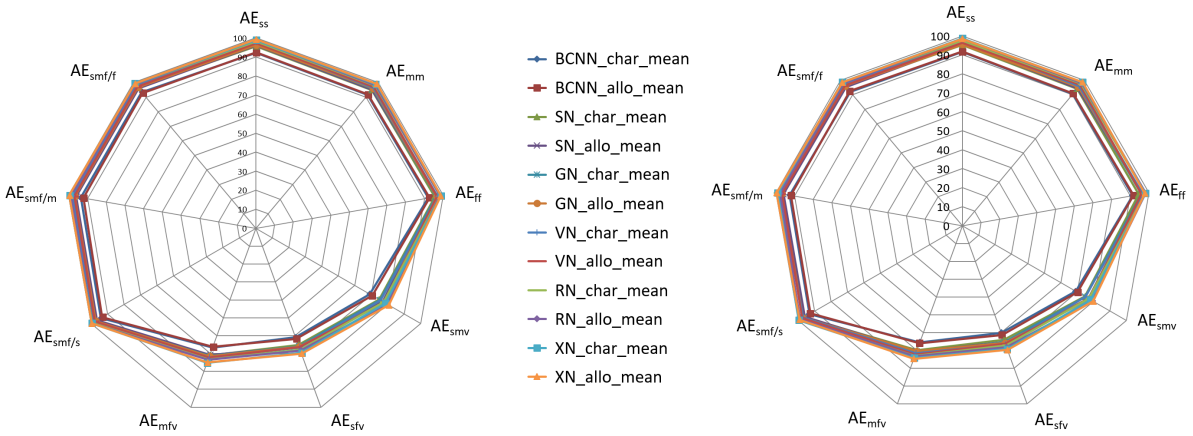


Figure 4.12: Radar plot of writer verification performance using auto-derived features. *Left*: representing Table 4.11 on D_c and *Right*: representing Table 4.12 on D_{uc}.

investigations with this network for increasing the 3-tuple (AE_{smv}, AE_{sfv}, AE_{mfv}) accuracies, and present them here and further in this chapter.

It can be reiterated that the 100 writers contributing to the D_c database are completely different from the 100 writers in D_{uc}. As a matter of fact, there is no writer overlap between D_c and D_{uc} (refer to *Section* 4.2.2).

We pre-trained the model using D_{uc} and repeated our experiments on D_c (say, E_D_{uc}/D_c). We adapted this idea from the *transfer learning* approach [208]. For pre-training, the total database D_{uc} was used. Then previous experimental setups, i.e., E_{sm}, E_{ms}, E_{sf}, E_{fs}, E_{mf} and E_{fm}, were used to obtain AE_{smv}, AE_{sfv}, AE_{mfv} accuracies (refer to *Section* 4.8.2). For example, we pre-trained the model using all the data of D_{uc}, now for the E_{sm} setup, we again trained the model with S_{s1} and tested on S_{m3}.

Likewise, the E_D_c/D_{uc} experiment was performed where the total D_c was used for pre-training, and then the earlier experiments were repeated on D_{uc} (refer to *Section* 4.8.2, 4.8.3). Here, two databases assisted each other in pre-training/learning to study the system performance. Such a technique may be referred to as *cross-learning*.

In Table 4.13, we present the Top-1 *writer identification* performances on E_D_{uc}/D_c

Table 4.13: Top-1 writer identification by pre-training

Setup	***Model*** (WI)	***3-tuple Accuracy (%)***			***Setup***	***Model*** (WI)	***3-tuple Accuracy (%)***		
		AE_{smv}	AE_{sfv}	AE_{mfv}			AE_{smv}	AE_{sfv}	AE_{mfv}
E_D_{uc}/D_c	XN_char_major	74.68	63.92	**68.09**	E_D_c/D_{uc}	XN_char_major	74.03	63.89	67.75
	XN_allo_major	73.64	63.38	**68.47**		XN_allo_major	72.78	63.24	67.54
	XN_char_mean	76.31	66.38	71.03		XN_char_mean	75.70	65.41	70.87
	XN_allo_mean	77.37	67.21	71.95		XN_allo_mean	76.57	66.56	71.76

and E_D_c/D_{uc} experimental setups employing various *Xception Net* models.

From Table 4.13, it can be observed that substantially the XN_allo_mean performed the best for both E_D_{uc}/D_c and E_D_c/D_{uc} setups. Overall, on 3-tuple accuracy, the highest-to-lowest performance order is as follows: XN_allo_mean $\succ$ XN_char_mean $\succ$ XN_char_major $\succ$ XN_allo_major. The only exception is to obtain AE_{mfv} accuracy in E_D_{uc}/D_c, where the XN_allo_major worked better than XN_char_major.

Overall, for E_D_{uc}/D_c and E_D_c/D_{uc} experiments, the Top-2 (Top-5) writer identification criteria produced up to 1.86% (7.52%) and 2.08% (8.37%) additional accuracy, respectively.

The *writer verification* performance for the experimental setups E_D_{uc}/D_c and E_D_c/D_{uc} are presented in Table 4.14. Here, for both the E_D_{uc}/D_c and E_D_c/D_{uc} experiments, mostly $\text{patch}_{\text{allo}}$ worked better than $\text{patch}_{\text{char}}$. The only exception is to obtain AE_{smv} in E_D_c/D_{uc} where XN_char_mean worked better than XN_allo_mean.

Table 4.14: Writer verification by pre-training

Setup	***Model*** (WV)	***3-tuple Accuracy (%)***		
		AE_{smv}	AE_{sfv}	AE_{mfv}
E_D_{uc}/D_c	XN_char_mean	83.05	72.21	76.82
	XN_allo_mean	83.54	72.45	77.65
E_D_c/D_{uc}	XN_char_mean	**82.58**	71.69	76.25
	XN_allo_mean	**82.34**	72.47	77.76

Compared to the results of *Section* 4.8.2 and 4.8.3, here it can be seen that such pre-training with cross-learning has improved the system performance (at most 5.23% for identification and 3.00% for verification).

Now, comparing the outcomes of experimental setups of Tables 4.13 and 4.14, we observed that E_D_{uc}/D_c performed better than E_D_c/D_{uc} for both identification and verification. Here also, all the models/experimental setups were compared using the *Borda count* [170].

4.8.6 Writer Identification/Verification on Enlarged Writer Set

In this section, we would like to see the system performance on an increased number of writers. For this purpose, we merged the controlled (D_c) and uncontrolled (D_{uc}) databases and get data from 200 writers. This experimental setup is denoted as E_D_{c+uc}.

Here also, we show the 3-tuple accuracies of *Xception Net*-based models as in *Section* 4.8.5, although we have computed 9-tuple accuracies from all models (refer to *Section* 4.8.2, 4.8.3).

The earlier experimental setups, i.e., E_{sm}, E_{ms}, E_{sf}, E_{fs}, E_{mf} and E_{fm} setups, were used here to obtain AE_{smv}, AE_{sfv}, AE_{mfv} accuracies (refer to *Section* 4.8.2). For example, the setup E_{sm} of E_D_{c+uc} was trained by the $S_{s1}+S'_{s1}$ set and tested on the $S_{m3}+S'_{m3}$ set.

In Table 4.15, we present the Top-1 *writer identification* performance in terms of 3-tuple accuracy, which shows that XN_allo_mean performed the best. Overall, on 3-tuple accuracy, the performance from highest to lowest order is as follows: XN_allo_mean ≻ XN_char_mean ≻ XN_char_major ≻ XN_allo_major. However, there is an exception for AE_{sfv} where XN_allo_major has worked better than XN_char_major.

Table 4.15: Top-1 writer identification on enlarged writer set

Model ($WI\_E\_D_{c+uc}$)	***3-tuple Accuracy (%)***		
	AE_{smv}	AE_{sfv}	AE_{mfv}
XN_char_major	73.96	**63.84**	67.27
XN_allo_major	72.72	**64.02**	67.16
XN_char_mean	76.07	66.37	70.79
XN_allo_mean	77.76	66.72	70.95

Overall for E_D_{c+uc}, the Top-2 and Top-5 writer identification criteria produced up to 0.47% and 4.38% additional accuracies, respectively.

In Table 4.16, we present the *writer verification* performance of the E_D_{c+uc} setup. In this case, XN_allo_mean worked better than XN_char_mean.

Table 4.16: Top-1 writer verification on an enlarged writer set

Model ($WV\_E\_D_{c+uc}$)	***3-tuple Accuracy (%)***		
	AE_{smv}	AE_{sfv}	AE_{mfv}
XN_char_mean	83.68	72.65	77.19
XN_allo_mean	84.02	72.81	77.87

For writer identification/verification on E_D_{c+uc}, XN_allo_mean performed the best. Here, all the models were compared using *Borda count* [170].

By comparing with the results from *Section*s 4.8.2 and 4.8.3, we noted that the system performance slightly increased (at most 4.02% for identification and 3.23% for verification) with the number of writers.

4.8.7 Comparison

To the best of our knowledge and understanding, our work is the earliest attempt of its kind on such a problem. Also, we did not find any other published research work on this topic to be compared.

4.9 Summary

In this chapter, we work on writer identification/verification when there is extensive variation in a person's handwriting. In brief, we focus on high intra-variable handwriting-based writer investigation. We employ both handcrafted and auto-derived feature-based models to study writer identification/verification performance. We generated two offline Bengali intra-variable handwriting databases from two different sets of 100 writers. For this database generation, we have also worked with auto-derived feature-based grouping technique to form similar groups of intra-variable writing. After experimenting on our databases, we observe that by training and testing on similar writing variability, our system produces encouraging outcomes. However, our system performance is comparatively lower for training and testing on disparate types of handwriting variability. We also attempt with cross-learning and see that the system performance improves with pre-training.

Here, a practical scenario is imitated, whereby a certain writing style of an individual is unknown (i.e., absent during training), and we note that the state-of-the-art methods do not perform well. However, we also observe that the deep features have a high potential for this task. In the future, we will try to exploit this potential and find some latent characteristics of a person from his/her varying styles of writing.

CHAPTER 5

IMPACT OF STRUCK-OUT TEXT ON WRITER IDENTIFICATION

Writer identification is a challenging task in the field of handwriting analysis owing to the intensive variation of human writing styles over space and time, as mentioned earlier in *Chapter* 1. However, promising results with acceptable accuracy have been obtained in the state-of-the-art methods of identifying a writer by his/her free-form running handwriting in various Roman-based western [144, 186] as well as Oriental scripts [44, 259]. A detailed survey on writer identification up to the year 1989 can be found in [186]. Some advanced information on this topic is also available in [144]. We have also reviewed the literature in *Section* 1.4.1 of *Chapter* 1.

Writer identification can be seen as a classification problem, where the task is to detect a writer class among n number of classes/writers, given a handwriting specimen. This specimen may be a full page of writing. Even a paragraph/text-line [258]/word [56]/character [41, 44] may be available for writer identification.

The recent writer identification techniques are built upon on edge/contour-based features [146, 181] of writing strokes, texture patterns [64], projection profiles [75], allographs [149], and combinations of various micro and macro features [216], etc. However, most published papers on writer identification consider ideal writing as the input, i.e.,

---

This work is partially published as:
C. Adak, B. B. Chaudhuri, M. Blumenstein, "Impact of Struck-out Text on Writer Identification", Proc. 30th Int. Joint Conference on Neural Networks (IJCNN), pp. 1465-1471, Anchorage, Alaska, USA, 14-19 May, 2017.

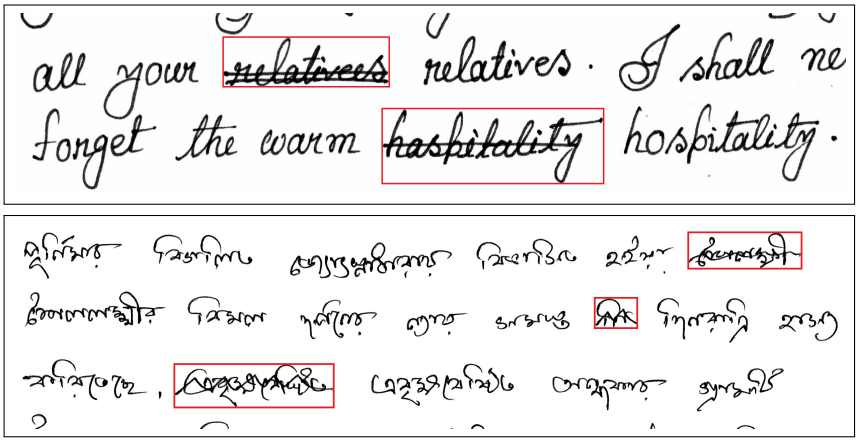


Figure 5.1: Examples of (a) English and (b) Bengali handwritten documents containing struck-out texts, marked by red boxes.

texts containing no writing error. In reality, a writer may strike-out/cross-out inappropriate words. Therefore, a handwritten document may contain such struck-out texts. Such examples are shown in Figure 5.1. The struck-out texts may have some impact on the automatic writer identification process, which is the primary focus of this chapter. For this purpose, we analyze the task, on struck-out text and normal text separately.

To perform the above, the detection of struck-out/crossed-out text is necessary. In the literature, a few papers deal with struck-out text detection. An early report by Arlandis et al. [109] mentioned “crossing-outs”, “scribbles”, “isolated strokes”, without giving any clear solution. Tuganbaev and Deriaguine [70] filed a US patent for their crossed-out character recognizer using a feature-based classifier. A work on HMM (*Hidden Markov Model*)-based crossed-out word recognition was reported in [139]. A graph-based approach for finding struck-out (or, “strike-through”) words from handwritten manuscripts was shown in [43]. A modified version of [43] was presented in [36], where a feature-based SVM classifier was used for struck-out text detection.

In this chapter, we consider offline handwriting of an alphabetic script *English* and an alpha-syllabary script *Bengali*. Recent advances in writer identification on Indic scripts are mentioned in [44]. Here, for struck-out text detection, we use CNN (Convolutional Neural Network) as a feature extractor and SVM (Support Vector Machine) as a classifier. For writer identification, we use two separate systems: (a) hand-crafted feature-based SVM classifier, (b) CNN with a recurrent neural network.

The main contributions of this chapter are as follows.

(i) Analyzing the impact of struck-out text on writer identification.

(ii) Detecting struck-out text by a hybrid model (CNN and SVM).

(iii) Identifying the writer using CNNs followed by a recurrent neural network.

(iv) Experimenting on alphabetic English and alpha-syllabary Bengali scripts as well as generating a database of handwritten specimens containing struck-out texts of 100 English and 100 Bengali writers.

The rest of the chapter is arranged as follows. *Section* 5.1 describes the proposed method. Then, the experiments and performance evaluation are presented in *Section* 5.2. Finally, *Section* 5.3 summarizes the chapter.

5.1 Proposed Method

As stated earlier, the impact of struck-out text on writer identification can be analyzed by identifying such text in the document. The work-flow of the proposed method for undertaking this is shown in Figure 5.2.

In the preprocessing stage, a handwritten page is segmented into text components using single-pass connected component labeling [87]. This single-pass algorithm is relatively faster than classical two-pass connected component labeling methods. Very small-sized components such as dots, dashes, commas, colons, etc. and noise are filtered out. We also segment the words using an off-the-shelf 2D Gaussian filter-based method, called *GOLESTAN-a* [166].

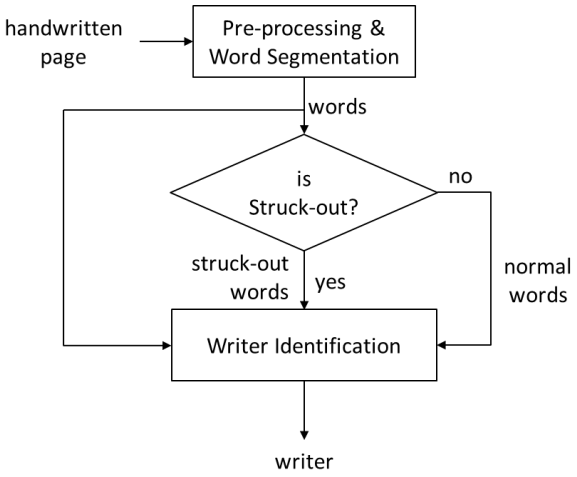


Figure 5.2: Work-flow of the proposed method.

5.1.1 Normal and Struck-out Text Separation

In handwritten documents, the ideal text without any writing-error is considered as "*normal*" text and the erroneous text containing a strike-out stroke is called as "*struck-out*" text. Here, the task is perceived as a binary classification problem, where we have to classify a word into *normal* versus *struck-out* class.

In recent days, the deep neural networks have created a benchmark for many machine vision applications [124]. The researchers have obtained outstanding results using CNNs as a feature extractor [254]. Sometimes a hybrid model performs better than stand-alone methods. Niu and Suen [249] proposed a CNN-SVM hybrid model for recognizing English handwritten numerals and obtained impressive results. In that work, they used a simplified CNN model instead of LeNet-5 [254].

5.1.1.1 Feature Extraction

In our approach, we adapt the common LeNet-5 [254] CNN architecture as per our requirement for feature selection. Our CNN model, employed as a feature extractor, is schematically shown in Figure 5.3.

Here, the preprocessed word level text components are fed into the CNN as inputs. These components are normalized to a fixed size of 32×92 pixels. The normalized image passes through convolutions followed by subsampling operations.

At this point, the first convolution layer C1 contains 6 feature maps, whereby a 5×5 convolutional filtering is used. Thus, each unit of the feature map is connected with a 5×5 area of the input image, called a "receptive field". All units in the feature map share the same kernels and the same set of weights. The size of this C1 feature map is 28×88.

The following subsampling layer S2 is comprised of 6 feature maps of size 14×44. Each feature map is connected with non-overlapping 2×2 receptive fields of C1. Here, a

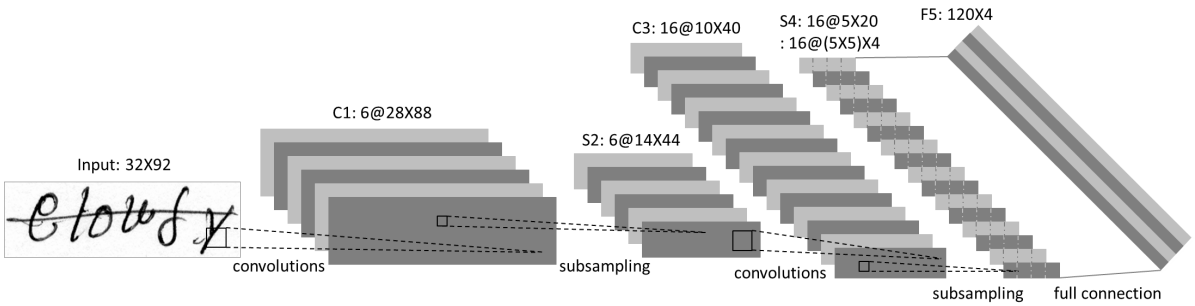


Figure 5.3: Our CNN architecture as a feature extractor.

max-pooling operation with 2×2 filter and a stride of 2 are used for the subsampling from C1 to S2.

The second convolution layer C3 is of 16 feature maps with a size of 10×40. Here also, a 5×5 filter is used, while each unit of the feature map is connected with 5×5 receptive fields of S2.

Similarly, C3 is followed by another subsampling layer S4. This S4 layer contains 16 feature maps of size 5×20. As before, the max-pooling operation is employed with a 2×2 filter size and a stride of 2.

Adapting the idea of LeNet-5, we add another layer F5 after S4, quite similar to the C5 of LeNet-5. In C5, LeCun et al. [254] used 120 feature maps. In LeNet-5, there was a full connection between S4 and C5, since S4 contained 16 feature maps of size 5×5 and the convolution kernel size was also 5×5. For our case, the S4 feature map size is 5×20. We assume it as 4 horizontally segmented non-overlapping feature maps of size 5×5 each. Therefore, we consider the F5 layer as 4×120 feature maps, where each 120 feature map is fully connected with the 5×5 segmented feature maps of S4.

In this way, our CNN model extracts the feature vector of dimension 480 (= 120×4).

5.1.1.2 SVM Classifier

The feature vector extracted from the CNN is to be fed into a classifier for separation of normal and struck-out texts.

We use the SVM with an RBF (Radial Basis Function) kernel [241] as a classifier, since it works better than MLP (Multi-Layer Perceptron), MQDF (Modified Quadratic Discriminant Function), kNN (k-Nearest Neighbors) and SVM-linear for handwriting analysis [36, 57]. Tuning of the SVM-RBF hyper-parameters (γ and $\mathscr{C}$) is required to avoid over-fitting and to control the decision boundary [129]. Such parameters are selected from a tuning set for the optimal performance of the classifier. This process is called "model selection". We use a traditional *grid-searching* technique for model selection. Here, k-fold cross-validation is also used. The choice of grid-searching range, selection of γ and $\mathscr{C}$, and the choice of k are discussed in *Section* 5.2.2.

5.1.2 Writer Identification

To see the impact of struck-out text on writer identification, we execute our experiments on three types of texts: (a) all handwritten texts, i.e., normal and struck-out, (b) only normal texts, (c) only struck-out texts.

The writer identification task can be perceived as an n-class classification problem, where the job is to map an unknown class (handwritten specimen) to its writer class-id.

In the pre-processing stage, we have segmented the handwritten page into words. We input these words, due to their discriminatory power [56], to the writer identification model.

A set of words per writer is used for training. At the testing phase, we also provide a set of words of an unknown writer. Each word is tested by the classifier separately and the classifier output is combined using a *majority voting* scheme. The choice of the number of words in the training and test set depends on the user with respect to the available data and the tool/classifier used. It may be of a predetermined fixed size or the number of words present in a full handwritten page. The details of the training and test sets are discussed in *Section* 5.2.2.2.

5.1.2.1 Hand-crafted Features with SVM Classifier

The traditional hand-crafted features are not derived automatically. In the literature, it has been seen that among such hand-crafted features, the contour-based features [146, 149, 181] work quite well for writer identification. For this reason, we choose the "contour-hinge" feature, proposed by Bulacu and Schomaker [149]. This feature tracks the orientation and curvature of the ink-strokes scribed by individual writers. In [149], the authors used the number of histogram bins $n_b = 12$ in a 360^o orientation span and obtained $n_b(2n_b + 1) = 300$-dimensional feature vector. We set $n_b = 16$, leading to a feature vector of dimension 528.

This feature vector is sent to an SVM classifier with an RBF kernel for classification/writer identification. We have discussed the SVM-RBF classifier in *Section* 5.1.1.2. Additionally, we use *one-against-all* strategy for this SVM due to multi-class/writer classification/identification. The hyper-parameter tuning and classifier training details are described in *Section* 5.2.2.2.

5.1.2.2 CNN-extracted Features with Recurrent Neural Network

We extract the word-patch level features automatically derived from the CNN as described in *Section* 5.1.1.1.

This feature vector of dimension 480 is provided as an input in a bi-directional *Recurrent Neural Network* (RNN) [157]. The number of nodes in the RNN input layer is

the dimension of the feature vector, i.e., 480. The number of individual writer classes is the number of nodes of the output layer. One ϵ node at the output layer is kept extra and remains as null. We use two distinct hidden layers for the forward and backward sequences, separately. Here, LSTM (Long Short-Term Memory) [206] blocks are used as hidden units. These two hidden layers contain 256 and 128 LSTM memory cells, respectively. This Bidirectional LSTM (BLSTM) recurrent net prevents the occurrence of so-called "vanishing gradient problem" [14]. All the meta-parameters are tuned and optimized using a tuning set, discussed in *Section* 5.2.2.2.

5.2 Experiments and Discussion

In this section, we present the experimental results and performance of our system. At first, we discuss the database employed for our experiments.

5.2.1 Database Employed

For experimental analysis, we required a database of handwritten documents containing struck-out texts. Among the publicly available databases, only the dataset used in [36] (henceforth called *NewISIdb: SoT*) contained a fair amount of such texts. However, for this study, we needed a reasonable amount of handwritten specimens of a particular writer for training and testing. The *NewISIdb: SoT* did not contain multiple copies of the handwriting of each individual and sometimes the writer information was absent due to the different motivation of their work. Therefore, it was necessary to generate a new database containing multiple copies of handwriting.

We collected English handwriting from 100 writers and Bengali writing from 100 individuals. Those writers were of both genders in the age group of 10 to 66 years with various academic backgrounds from elementary school to university level. Each writer provided two full-pages of handwriting samples. The handwritten content was independently chosen by the writer from any piece of article. The writers were requested to strike-out some words in their running handwriting styles. It was guaranteed that the handwritten pages contained some struck-out texts. We supplied $A4$ sized 75 GSM (g/m^2) blank white pages and a 0.5 mm ball-point black-ink pen of the same brand/model in order to maintain the consistency with respect to color and stroke-width.

Our database contained a total of 200 English and 200 Bengali handwritten pages. Here a handwritten page contained approximately 16 text-lines and a text-line contained

about 10 words. Therefore, a total of almost 3200 (= 200×16) text-lines and 32000 words were available for both English and Bengali scripts. For each writer, about 320 handwritten word samples were present. Precisely, the total numbers of struck-out words and normal words were 2152 (2317) and 30045 (29341), respectively, for the English (Bengali) script in our database. Here, the *strike-out rate*s[1] [36] were 6.68% for English and 7.31% for Bengali script.

5.2.2 Results and Evaluation

In this section, we present the experimental results and performance of our method for struck-out text detection and their effect on writer identification.

5.2.2.1 Normal and Struck-out Text Separation using CNN-SVM

This step was like a pre-processing stage of our main task of analyzing the impression of struck-out text on writer identification. However, we also wanted to evaluate the performance of our CNN-SVM hybrid classifier for such normal versus struck-out class detection.

Besides our generated database (*Section* 5.2.1), we also tested our method on the *NewISIdb: SoT* database [36]. The *NewISIdb: SoT* contained a total of 1395 (1432) struck-out and 30670 (32762) normal English (Bengali) words.

For training purposes, we used 50% data of our generated database and 20% of the *NewISIdb: SoT*. For the CNNs, the training epochs were increased up to 500. The CNN was trained with stochastic gradient descent. We employed a learning rate of 10^{-3} with a momentum term of 0.9 for this CNN. The SVM-RBF hyperparameters (γ and $\mathscr{C}$) were also tuned by a tuning set. The grid searching range for γ and $\mathscr{C}$ were $[2^3, 2^2, \ldots, 2^{-7}]$ and $[2^7, 2^6, \ldots, 2^{-4}]$, respectively. The best performance was obtained for $\gamma = 2^{-3}$ and $\mathscr{C} = 2^6$. Here, we used 5-fold cross-validation.

The performance of struck-out text detection is calculated in terms of *Precision* (P), *Recall* (R), and *F-Measure* (FM).
True Positive (TP) := # genuine struck-out words detected by our system.
False Negative (FN) := # genuine struck-out words incorrectly recognized as normal.
False Positive (FP) := # normal words detected as struck-out.

[1] $strike\text{–}out\ rate = \frac{\#struck\text{–}out\ words}{\#total\ words}$.

Table 5.1: Struck-out text detection performance

Script	*English* (*Bengali*)		
Database	***Precision %***	***Recall %***	***F-Measure %***
Generated	98.45 (98.16)	98.93 (98.67)	98.69 (98.41)
NewISIdb: SoT [36]	98.63 (98.25)	99.08 (98.84)	98.85 (98.54)

$$P = \frac{TP}{TP+FP}\ ; \quad R = \frac{TP}{TP+FN}\ ; \quad FM = \frac{2\times P\times R}{P+R} \tag{5.1}$$

The quantitative performance is shown in Table 5.1.

For English (Bengali) struck-out text detection, we obtained 98.69% (98.41%) and 98.85% (98.54%) of F-Measure on our generated database and *NewISIdb: SoT*, respectively.

5.2.2.2 Struck-out Text Impression on Writer Identification

As stated earlier, our writer identification method was tested on three categories of data: (a) full handwritten page (normal + struck-out texts), (b) normal texts only, (c) struck-out texts only.

For writer identification, we employed the Top-N criterion, where the possible writer was a member of a reduced set of 'N' ($\ll$ total number of writers) individuals. Here, we chose Top-1, Top-2, and Top-5 criteria and provided the performance in terms of *accuracy*.

5.2.2.2.1 Writer Identification using Hand-crafted Feature-based SVM : In our generated database, each writer wrote two full-pages containing some struck-out texts in their natural writing style. We used one page for training and the other page for testing. We did not use any fixed size word vocabulary for training, since we intended to make the system quite independent of word segmentation. Therefore, if the words were not properly segmented, then our method would still perform well. Here, one page having approximately 160 words was used for training, per writer. The hyper-parameters (γ and $\mathscr{C}$) for SVM-RBF were tuned using a tuning set. The best performance was obtained

Table 5.2: Writer identification performance using SVM

Script	*English* (*Bengali*)		
Handwritten Text	***Accuracy %***		
	Top-1	*Top-2*	*Top-5*
Normal + Struck-out	69.27 (68.26)	70.75 (69.87)	71.72 (71.23)
Normal	74.85 (73.62)	77.46 (76.37)	79.62 (79.08)
Struck-out	25.57 (24.20)	26.42 (24.81)	26.77 (26.63)

for $\gamma = 2^{-4}$ within the range $[2^3, 2^2, \ldots, 2^{-8}]$ and $\mathscr{C} = 2^7$ within the range $[2^8, 2^7, \ldots, 2^{-2}]$. Here, 5-fold cross-validation was used.

The writer identification performance employing a hand-crafted feature-based SVM in terms of *accuracy* is presented in Table 5.2.

In Table 5.2, the Top-1 criterion produces an overall 69.27% (68.26%) accuracy on full-page English (Bengali) writing, while it is 74.85% (73.62%) when tested on the normal text. It can also be observed that removal of struck-out text increases the performance of writer identification. Here, Top-1 accuracy has increased by 5.58% for English and 5.36% for Bengali script.

5.2.2.2.2 Writer Identification using Auto-derived CNN Feature-based RNN : Here also, we used one page per writer for training as in the procedure of *Section* 5.2.2.2.1. We employed a learning rate of 10^{-3} with a momentum term of 0.9 for this neural net. The training epochs were increased up to 500.

In Table 5.3, we present the writer identification performance in terms of *accuracy* employing an automatically extracted CNN feature-based recurrent net.

Table 5.3 shows that our writer identification method performs 3.34% (3.27%) better with respect to the Top-1 accuracy on normal English (Bengali) text than the mixed-up text.

The writer identification performance on struck-out text alone is very poor. The possible reasons for such poor performance are as follows.

(i) The dataset contains a very small amount of struck-out texts per writer. Since our dataset is quite realistic and contains natural running-form of handwriting, the struck-

Table 5.3: Writer identification performance using CNN-RNN

Script	*English* (*Bengali*)		
Handwritten Text	***Accuracy %***		
	Top-1	*Top-2*	*Top-5*
Normal + Struck-out	84.14 (83.64)	84.70 (84.41)	86.69 (85.66)
Normal	87.48 (86.91)	88.43 (88.13)	89.27 (89.25)
Struck-out	29.97 (28.52)	30.82 (30.14)	32.58 (31.23)

out rate becomes 6.68% (7.31%) for English (Bengali) pages. Therefore, the reduced amount of data for training struck-out text may have reflected the performance while identifying writers on struck-out texts only.

(ii) Sometimes, the strike-out strokes damage the writing strokes heavily. Therefore, the discriminative power of writing strokes diminishes and performs poorly for writer identification.

Our method confirms that struck-out text impedes the writer identification performance, while the normal text plays a more important role in this task.

5.2.3 Comparison

Among previous works for struck-out text detection, the method of [109] did not provide any solution. Tuganbaev and Deriaguine [70] dealt with struck-out (or, "stricken-out") characters only. Being a US patent [70], proper technical detail for implementation was missing there. In [139], the straight and wavy strokes used to strike-out were machine-generated and superimposed on offline words. In our case, the strike-out strokes were generated by humans at the time of writing. Only the methods of [36, 43] were related to our task of struck-out word detection. The system of [43] worked with only straight-line type strike-out strokes, while the updated version [36] worked with various forms of strike-outs. Therefore, we compare our struck-out text detection method with the scheme of [36] on the same database (*NewISIdb: SoT*) as employed in [36]. This quantitative comparison in terms of *Precision*, *Recall* and *F-Measure* (*FM*) is presented in Table 5.4. On the *NewISIdb: SoT*, for struck-out text detection, our method acquires 98.85% (98.54%) of *FM* on English (Bengali), whereas the method of [36] produces an *FM* of 91.56% (91.06%).

Table 5.4: Comparison of struck-out text detection

Script	*English* (*Bengali*)		
Method	***Precision %***	***Recall %***	***F-Measure %***
Method of [36]	90.94 (90.19)	92.18 (91.94)	91.56 (91.06)
Proposed method	98.63 (98.25)	99.08 (98.84)	98.85 (98.54)

We found that the work of Brink et al. [7] was related to our task. To separate the struck-out/"crossed-out" text, they employed a *decision tree* classifier using two simple hand-crafted features based on crossing counts of writing strokes (*"branching"*) and ink-pixel counting (*"size"*). Their method removed 47.5% of struck-out text retaining 99.1% of the normal text, while tested on a real forensic dataset of the *Netherlands Forensic Institute* (NFI). We contacted the group leader of [7] who informed that the NFI biometric data were taken from real criminal suspects and it could not be shared with us. Therefore, we were not able to compare our results on the NFI dataset. However, our method detected 98.69% (98.41%) and 98.85% (98.54%) struck-out English (Bengali) text in terms of F-Measure while testing on our generated database and *NewISIdb: SoT*, respectively. After removal of struck-out text, the writer identification performance of [7] deteriorated by about 1%, while in our case, this performance improved by approximately 3%–5%. The method of Brink et al. [7] removed some good normal texts as struck-out/"crossed-out", whereas our method was very precise about normal and struck-out text separation.

A slightly related work in [112] discussed the effect of ruling-line deletion on writer identification. They reported that by retaining rather than by deleting the ruling-lines increased the writer identification performance. However, the strike-out strokes were hand-drawn irregular patterns (straight horizontal, slanted, crossed, wavy, zigzag, etc.), whereas ruling-lines were of a regular pattern, i.e. underline-like, pre-printed straight-lines. The ruling-line affected most of the words of a page, while the struck-out rate of our database was 6.68% (7.31%) for English (Bengali). Therefore, retaining the strike-out strokes, like retaining the ruling-line, had no positive impact on writer identification. On the other hand, deleting the strike-out stroke and modifying the writing stroke using the method of [36] were computationally costly, and also were not so effective on the writer identification performance.

5.3 Summary

In this chapter, we investigate the effect of struck-out texts on the automated writer identification process. We show that the presence of struck-out texts impedes writer identification performance. A CNN-SVM hybrid model is used to detect struck-out texts. We employ both hand-crafted and auto-derived features for writer identification. The hand-crafted features are fed into an SVM classifier, and the auto-derived features are supplied to an RNN model. We generate a handwritten document database containing some struck-out texts from 100 English and 100 Bengali writers. We have obtained a 98.85% (98.54%) of F-Measure for struck-out text detection on English (Bengali) handwriting while testing on this database. Here, for English (Bengali) handwriting, the presence of struck-out texts degrades the writer identification accuracy by 5.58% (5.36%) and 3.34% (3.27%) while employing the hand-crafted and auto-derived features, respectively.

Although the struck-out text removal and processing only on normal texts improve the writer identification a little (about 3%–5% accuracy), our method analyzes that the same features used for normal texts and struck-out texts drop the performance. For a page, having a higher strike-out rate, i.e., containing relatively more struck-out texts and less normal texts, the writer identification performance becomes lower. Therefore, for such cases, the general writer identification technique does not work well. However, auto-derived features may perform better for struck-out texts on the availability of more struck-out training data. Our next endeavor will be to analyze such cases where a page will contain approximately 15%–20% of struck-out texts/writing errors. Therefore, our future plan is to collect more struck-out data and perform the experiments again. Moreover, we believe that a writer retains his/her latent identity in strike-out strokes. We will also try to explore this in the near future.

Part II

Understanding Writing Strokes

CHAPTER 6

LEGIBILITY AND AESTHETIC ANALYSIS OF WRITING STROKES

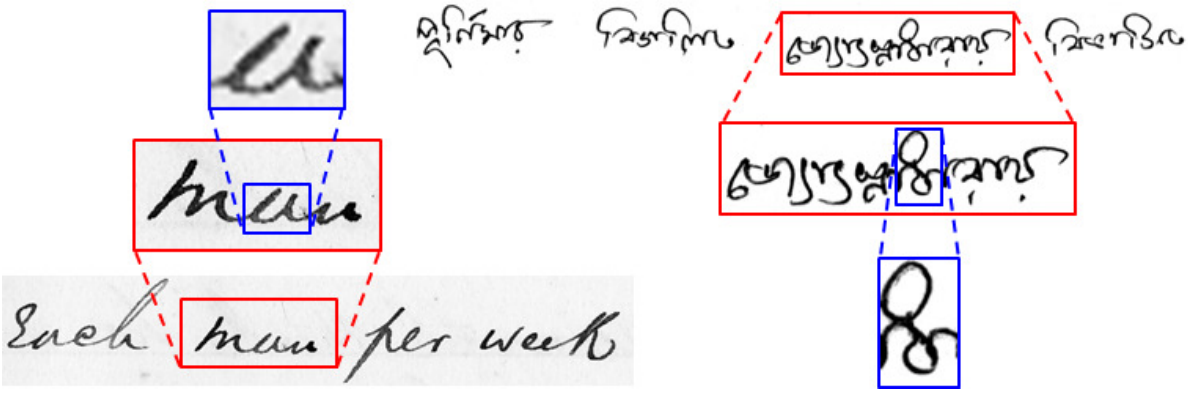
Handwriting is a concatenation of graphical symbols drawn by a pen or, other writing instruments by the hand in order to represent linguistic constructs for communication and knowledge storage. These graphical marks/writing symbols have deep orthographic relation to the phonology of a spoken language [173], as discussed earlier in *Chapter* 1. These writing symbols and orthographic rules evolve through the progress of civilization. The transformation has also occurred in writing medium, i.e., tool (from wedge to the pen) and sheet (from clay-tablet to the paper).

The significant purpose of handwriting is the communication between the reader and writer [76]. To achieve this purpose, the writer should know how to write the graphical symbols and the reader should know the phonology of these symbols. Furthermore, one basic criterion, required for proper communication, is the "legibility", that means "the quality of being clear enough to read"[1]. The *legibility of handwriting* is a well-known topic in the *Educational research* [151] domain for more than 100 years [69, 82, 143]. But, in automated *Document Image Analysis* (DIA) and in the *Handwriting Recognition* (HR) [187] domain, the legibility is relatively a new research issue.

Sometimes legibility is considered synonymous to the "readability" [143]. In Oxford

This work is partially published as:
C. Adak, B. B. Chaudhuri, M. Blumenstein, "Legibility and Aesthetic Analysis of Handwriting", Proc. 14th Int. Conf. on Document Analysis and Recognition (ICDAR), pp. 175-182, Kyoto, Japan, 9-15 Nov., 2017.

[1] `https://en.oxforddictionaries.com/definition/legibility` , retrieved on 31 Dec., 2018.



(a) (b)

Figure 6.1: Examples of (a) English and (b) Bengali readable texts having character-formation ambiguity.

dictionary, the meaning of "readability" is given as "the quality of being legible or decipherable"[2]. Readability is actually a higher level of intelligence. It includes legibility with an easy understanding of the semantically well-formed text. For example, if a person, who is well-versed in the English language, but not in Greek, can go through the Greek handwriting written in the similar English alphabet, but cannot comprehend its linguistic meaning. Therefore, for him/her there is void in readability, although the text is legible. We consider the legibility as a part of readability, and if a well-formed text is legible, then it is obvious that the text is readable, provided the reader has linguistic knowledge.

A human reader having usual knowledge of English cursive writing can easily read the text-line in Figure 6.1:(a) as "Each man per week". However, reading the red-boxed highlighted word alone is likely to be confused as 'm'+'u'+'n' or something else, instead of "man", since the blue-boxed character has shape ambiguity. Nevertheless, a human brain, trained with language/script vocabulary and grammar, can make use of neighboring characters/words to get over the ambiguity, and here the human cognition system perceives the word as "man". Similarly, Figure 6.1:(b) shows an example in Bengali script, where the red-boxed readable word contains the blue-boxed ambiguous character. In this chapter, although we focus mainly on the alpha-syllabary Indic script Bengali [37, 178], we show an example in alphabetic Latin script English in Figure 6.1:(a) for more readerships.

Actually, the legibility depends highly on the character scribbling that is based on

[2]`https://en.oxforddictionaries.com/definition/readability` , retrieved on 31 Dec., 2018.

personal writing style. The ambiguous character stroke formation may lead to poor legibility. Often, this formation of characters is considered as one of the main judgment criteria in the *excellence of handwriting* [82]. The character formation style may be broadly categorized into the *manuscript* and *cursive* [152] writing style. The manuscript writing, where each character is distinctly written, is typically more legible than cursive which is continuous [76].

The manuscript style for the English script is taught at the elementary school level and the cursive style is practiced before the end of primary grades [76]. The Bengali manuscript style is also trained in elementary school level. However, there is no common practice to transit from manuscript to cursive styles in the schools of countries with Bengali mother tongue such as Bangladesh and India. Still, the cursive style is observed in Bengali handwriting, possibly due to the influence of Muslim and British colonial rule [45], when Persian and English scripts dominated in office works. Now a person attains his/her own cursive style by observing the handwriting of teachers, friends, neighbors and some eminent persons, e.g., *R. N. Tagore*.

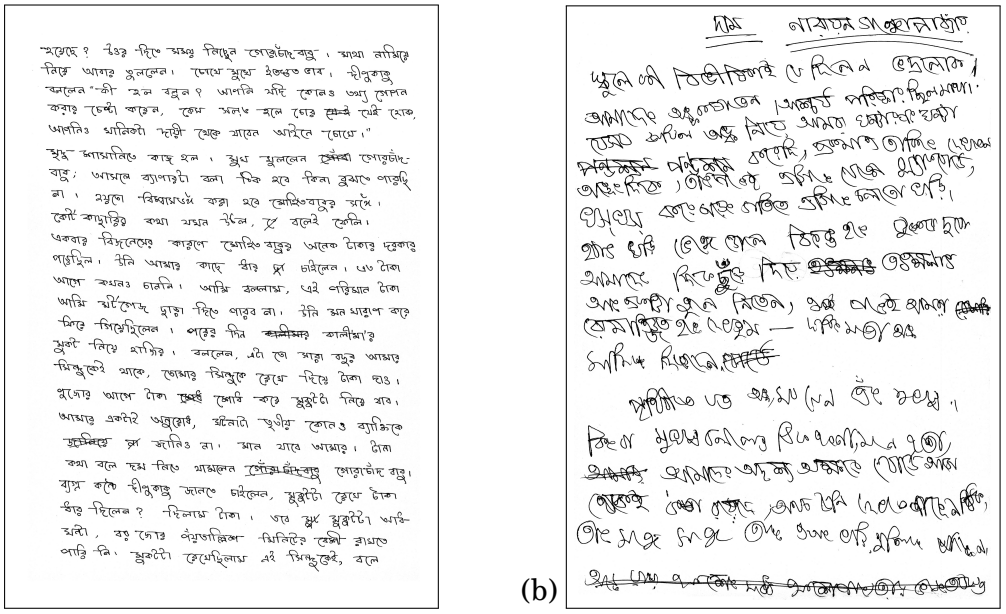
There are about 300 basic and conjunct characters in Bengali script [37]. Moreover, some conjunct characters can be written in more than one form [44], as discussed previously in *Chapter* 2. Even some educated person may write a complex conjunct character in a wrong way due to a lack of proper handwriting education in school. In such scenario, those conjunct characters may individually become illegible. Thus, the research on Bengali handwriting legibility analysis become more challenging owing to the inadequate writing practice in school, the variation of individual cursive writing styles, and the complex patterns of conjunct characters.

Besides legibility, sometimes the handwriting aesthetic receives the attention of the reader. As per the Oxford English dictionary, "aesthetic" is "concerned with beauty or the appreciation of beauty"[3]. The aesthetic study of the digital photographic image [179] becomes popular where the researchers attempt to find "beautiful attributes" [141]. In the DIA domain, such aesthetic analysis is also very new [23].

We consider "aesthetic" as a characteristic that gives the perception of beauty to the human cognitive system at the first glance, which may attract a person to read it. Generally, the aesthetic handwriting is more likely to be legible, if honestly written. However, the converse is not generally true. In Figure 6.2, we present a pair of examples of the handwritten document with the high and low aesthetic property.

The purposes of the automated analysis of handwriting legibility and aesthetic sense

[3] `https://en.oxforddictionaries.com/definition/aesthetic` , retrieved on 31 Dec., 2018.



(a) (b)

Figure 6.2: Examples of (a) high and (b) low aesthetic handwritten Bengali document.

are as follows.

(i) Attempting to impart the machine a human-like perception and to make an artificial cognitive system that understands the legibility and aesthetics of handwriting.

(ii) Using as a pre-processing module of an OCR (Optical Character Recognition) engine, for sorting the texts with respect to legible and aesthetic score. It is more likely that highly legible and more beautiful texts will show high recognition accuracy by OCR.

(iii) Analyzing the legibility issues of certain script-specific characters and its influence on the aesthetics of script. For example, people usually have the problem of writing some Bengali conjunct characters properly and beautifully.

Here, we formulate our tasks of legibility and aesthetic analysis as feature-based classification problems. For legibility analysis, the features are extracted automatically using CNN (Convolutional Neural Network). To analyze the aesthetics of handwriting, we use some hand-crafted features. The auto-derived and hand-crafted features are fed to RNN (Recurrent Neural Network) and SVM (Support Vector Machine) classifier, respectively. We employ several human volunteers to put some subjective score as ground-truth.

Our contributions in this chapter are as follows.

(i) Investigating legibility of handwriting stroke.

(ii) Examining the aesthetic value of the handwritten text.

(iii) Attempting to find a relatively new patch selection strategy without normalizing

the input image, for automated feature extraction using CNN.

(iv) Analyzing handwriting legibility using a hybrid neural net model (CNN and RNN).

(v) Generating a corpus of offline Bengali handwriting with legibility and aesthetic score set by human volunteers.

Very few somewhat related works on this domain exist in the literature of DIA. Chou and Yu [210] considered the quality of offline Chinese character for sorting the characters in a database. They used a feature-based Bayes' decision rule for this sorting. Three features were calculated from character skeleton: (a) stroke-density distribution in four directions, (b) histogram by horizontal and vertical projection, (c) stroke crossing-count in horizontal and vertical directions. It was an early attempt for character sorting in terms of character formation quality, somewhat related to legibility.

Majumdar et al. [23] worked with various coarse and fine level features for visual aesthetic analysis of offline English handwritten document. They employed a linear SVM to classify the documents into five classes, grouped by subjective aesthetic score. They obtained the coarse-level features from word bounding box height, word-spacing, horizontal profiles, and discrete Fourier transformation. Among fine-level features, they used connected component and stroke-based features, gradient and binary descriptor-based features, and texture features. For this fine-level feature extraction, they observed the "local properties of the writing", those were used in "somewhat similar to the degradation models" [23]. However, focusing on handwriting legibility and aesthetics, the assessment of document image quality [175, 239] is a different issue that is related to the degradation due to successive printing (writing), scanning and transmission.

In [234], the fractal behavior of writing strokes was employed to generate some synthetic parameters, with the objective of handwriting classification. With this target, Bouletreau et al. [234] used a parameter named "legibility rating of the image", which is some sort of "fractal dimension of writing". Any direct quantitative analysis of handwriting legibility is missing there due to a different motivation of their work. We did not find any other work fairly related to the automated legibility and aesthetic analysis of handwriting in the domain of DIA. In *Section* 6.2.3, we provide a comparative discussion with these few related works.

The rest of the chapter is organized as follows. *Section* 6.1 describes the proposed method. Then, the experiments and results are presented in *Section* 6.2. Finally, *Section* 6.3 summarizes the chapter.

6.1 Proposed Method

In this section, at first, we describe the problem formulation and then proceed to our planned methodology.

6.1.1 Problem Formulation

Both the legibility and aesthetic sense depend on human perception and here we try to teach the machine by this cognitive perceptivity.

Some texts in paragraph, word and character level are shown to the human readers and asked to put a subjective score of legibility within a continuous range of $[L_1,L_2]$; $L_1,L_2 \in \mathbb{R}$. Similarly, for aesthetic text analysis, the human readers are requested to provide some score in a continuous range $[A_1,A_2]$; $A_1,A_2 \in \mathbb{R}$. At the time of data collection, the subjective scores are put in a continuous range, since initially, we do not want to restrict the scores in a fixed discrete opinion scale such as *Likert scale* [185].

For each text, multiple persons provide their opinion scores. We calculate the arithmetic mean of all these scores and tag with each text as its *mean opinion score*.

After the data collection, the legibility score range $[L_1,L_2]$ is divided into n_L bins. Therefore, all the *legibility mean opinion score*s of all handwritten texts fall into these n_L bins. In other words, all the handwritten texts are categorized into n_L classes with respect to the legibility score. Similarly, the texts are annotated with *aesthetic mean opinion score*s spreading into n_A bins of score range $[A_1,A_2]$.

Thus, we map the handwriting legibility and aesthetic analysis task into classification problems. Now, the task is to label a given text into n_L-classes of legibility and n_A-classes of the aesthetic score.

6.1.2 Preprocessing

In the preprocessing stage, we label the components of a handwritten page by a relatively faster method of single-pass connected component labeling [87]. The text region is separated from the doodle/drawing-like non-text components, if any, by employing the technique in [42]. In the text region, the struck-out texts are also detected using the system of [36]. Very small sized components such as dot, comma, dash, colon, etc. and noise are eliminated. The text-lines and words are segmented by employing an off-the-shelf 2D Gaussian filter-based method *GOLESTAN-a*, as discussed in [166]. The

character level segmentation is done by the water reservoir principle-based method of [228].

6.1.3 Legibility Analysis

For legibility analysis, we require the character level information of a document. Here we employ a feature-based classification method for this analysis.

6.1.3.1 Feature Extraction for Legibility Analysis

For feature extraction, we rely on convolutional neural net-based automatically derived features. Here, we adapt common LeNet-5 CNN architecture [254] and use it with some modifications as per our requirements, such as the number of feature maps, convolution and sub-sampling operations.

CNN generally takes an input of a fixed size image. For character recognition task, resizing and normalizing the handwritten image work well as input to CNN. However, for a task where writing style holds major information, normalization may impede the performance [29]. In our legibility analysis job, we require the information regarding individual writing style. Therefore, we employ the patch-based scheme of [235] with some modification along with CNN architecture, as per our necessity.

For the CNN architecture, we have already obtained the character-level information from the pre-processing step. On a segmented character image, we find its center of gravity (p_{CG}). By centering the p_{CG}, we choose a 128×128 neighbor window and use this window as a character-level patch, input to the CNN.

We do not select the patch through the classical sliding-window technique, since the text-lines are not skew-normalized. Sliding a window through the horizontal median of the text-line main-body height could be performed, but there is a high possibility

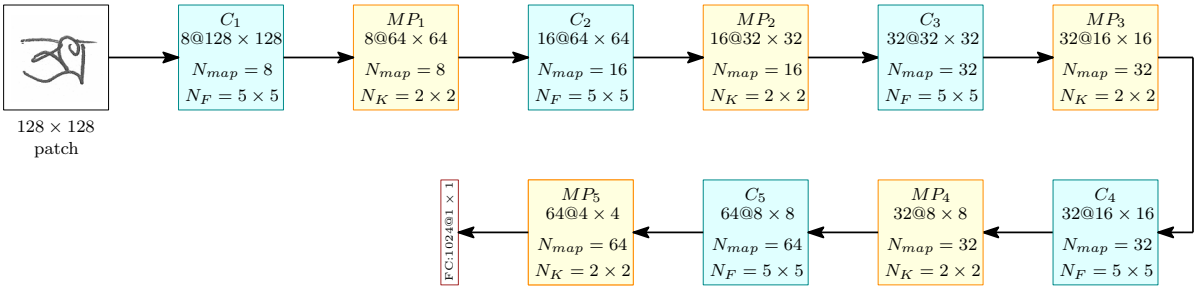


Figure 6.3: Our CNN architecture as a feature extractor.

of information loss for extremely skewed/wavy text-lines and less inter-text-line gaps. The patch-window size is chosen through experimental analysis. It may be noted that a good number of Bengali compound characters are more elongated in vertical than the horizontal direction, but we choose the character-level patch window as a square box. The reason is that we do not normalize the characters, and the character may not fit properly inside a specific sized box. Therefore, sometimes neighboring character components may enter into this box due to improper character segmentation and sometimes a portion of the character may remain out of this box. Thus, we come up with experimentally choosing a 128×128 square window.

This CNN model contains 5 convolutional layers, each followed by a sub-sampling (with *max-pooling* operation) layer. The filter size (N_F), kernel size (N_K) and the number of feature maps (N_{map}) for each of the convolutional layers (C_j, for $j = 1,2,\ldots,5$) and subsampling/max-pooling layers (MP_j, for $j = 1,2,\ldots,5$) for our CNN architecture are mentioned in Figure 6.3. We use ReLU as an activation function.

Our automated feature-extractor CNN model ends with a fully connected layer (FC) containing 1024 neurons and thus produces 1024 dimensional feature-vector F_{CNN}.

6.1.3.2 Classification for Legibility Analysis

The extracted features are to be fed into a classifier to assign a class on an unknown handwriting specimen, with respect to legibility score.

We employ a bidirectional *Recurrent Neural Network* (RNN) [157] for the automatically derived CNN-extracted features. The RNN input layer contains exactly the same number of nodes as the dimension of the feature vector, i.e. 1024. The total number of legibility-score classes denotes the number of nodes in the output layer. One ϵ node at the output layer is kept extra as null. Two distinct hidden layers for the forward and backward sequences are engaged. Here, LSTM (Long Short-Term Memory) [206] blocks are employed as hidden units. These two hidden layers comprise 512 and 256 LSTM memory cells, respectively. Such recurrent net prevents the occurrence of so-called "vanishing gradient problem" [14]. The combination of bidirectional RNN and LSTM is briefly called as BLSTM (Bidirectional LSTM) architecture [14].

6.1.4 Aesthetic Analysis

The aesthetic analysis is performed on the full page/document as a whole. We do not focus here on character/grapheme-level aesthetic study. Such aesthetic perception is

like the visual impact (attraction/distraction) at the first sight. The aesthetic analysis is tackled using a feature-based classification method.

6.1.4.1 Feature Extraction for Aesthetic Analysis

We design some empirical hand-crafted features for the aesthetic analysis of a document. These features are described as follows.

6.1.4.1.1 Density of Doodle / Drawing : A handwritten document may contain non-textual doodle or drawing by a pen/pencil. These doodles/drawings may influence the overall aesthetic perception of the document. Such non-texts are labeled in the pre-processing stage (refer to *Section* 6.1.2). We calculate the black density (d) of a doodle/drawing in a component bounding box (BB), as follows.

$$d = \frac{number\ of\ black\ pixels}{total\ number\ of\ pixels\ in\ a\ BB} \tag{6.1}$$

Multiple doodles may present in a handwritten page. We compute the black densities of all doodles. We also calculate the average density (d_{avg}) and standard deviation of the density (d_{SD}) and use it as feature f_1 and f_2, respectively. If there is no doodle/drawing present in the document, then we set both d_{avg} and d_{SD} as zeros. For the presence of only one doodle in the document image, the value of d_{SD} is zero.

6.1.4.1.2 Strike-out Rate : The presence of struck-out text is possible in a free-form handwritten document. We observe that such struck-out text has an impact on the aesthetics of handwriting. These struck-out texts are already detected in the pre-processing stage (refer to *Section* 6.1.2). In a document image, the strike-out rate [36] is calculated and used as feature f_3.

$$Strike\text{–}out\ rate(f_3) = \frac{number\ of\ struck\text{–}out\ words}{total\ number\ of\ words\ in\ a\ page} \tag{6.2}$$

6.1.4.1.3 Marginal Space Width : The overall aesthetic of a handwritten document depends on the margin space width and its variation. The proper boundary of a text gives a perception of the visual object inside a frame.

The top and bottom marginal white space widths from the page borders are calculated using horizontal projection profile [165], and are used as feature f_4 and f_5, respectively.

From the left border of the page, the marginal white space widths (w_{Li}) of the left-most word bounding box of each text-line are calculated. The average and standard

deviation of w_{Li} are calculated, and used as feature f_6 and f_7, respectively. Similarly, the average and standard deviation of right marginal white space widths (w_{Ri}) are computed to employ as feature f_8 and f_9, respectively.

6.1.4.1.4 Text-line Skewness : The text-lines, when mostly written in white blank pages, may be skewed from the standard horizontal axis. Such irregular skewness may influence handwriting aesthetics.

The text-lines are segmented in the preprocessing stage (refer to *Section* 6.1.2). Through the middle of the text-line, an imaginary straight line $\mathscr{L}$ is considered. The text-line skewness is measured by an angle between this imaginary line $\mathscr{L}$ and the horizontal axis. Each text-line may be skewed upward/downward making positive/negative angle from the horizontal axis. We calculate the average and standard deviation of all positive skew angles and use as feature f_{10} and f_{11}, respectively. Similarly, for the downward skewness, the average and standard deviation of all negative skew angles are computed and employed as feature f_{12} and f_{13}, respectively.

Sometimes, there may be waviness (ups and downs) in some text-lines. For such a wavy text-line, we construct the middle imaginary line by drawing a straight line which covers the text-line mostly, and then calculate the above features in a similar way.

6.1.4.1.5 Word Slant : The uniformity in word-slants is considered as a beautiful attribute of handwriting. The words have already been segmented in the pre-processing stage (refer to *Section* 6.1.2). The word slant is calculated by the angle between the vertical part of the character and the standard vertical axis. For the word slant calculation, we use the method of [74] which is based on vertical projection profiles and the *Wigner-Ville distribution* [77]. In a handwritten page, the average and standard deviation of the word-slants are calculated and used as feature f_{14} and f_{15}, respectively.

6.1.4.1.6 Inter-line and Inter-word Gap : A writer maintains a white-line gap between two consecutive text-lines, and the white-spaces among successive words. Such inter-line and inter-word gaps enhance the handwriting aesthetics. From the text-line and word segmentation information, these gaps are obtained at the preprocessing step (refer to *Section* 6.1.2). The average and standard deviation of the inter-text-line gaps in a page are used as feature f_{16} and f_{17}, respectively. Similarly, the features f_{18} and f_{19} are obtained from the average and standard deviation of inter-word gaps, respectively.

6.1.4.1.7 Text Main-body Height : The regularity in text main body height beautifies the handwritten text. Here, we actually calculate the main-body height of an individual word. The Bengali words can be partitioned into three zones: upper, middle and lower [37], by two imaginary lines. These two lines can be obtained by horizontal projection profile. The middle zone is considered as text main body region, where the maximum number of black pixels resides and the horizontal histogram reaches global maximum [231]. We compute the height of the main body of each word in a page. The average and standard deviation of these text-heights are used as feature f_{20} and f_{21}, respectively.

In this way, we obtain these 21 single-valued features from a handwritten document image.

6.1.4.2 Classification for Aesthetic Analysis

The extracted features are supplied to an SVM classifier to categorize a document according to its aesthetic score.

To work with the hand-crafted features, we use the SVM with an RBF (Radial Basis Function) kernel [241] as a classifier, since it works better than MLP (Multi-Layer Perceptron), kNN (k-Nearest Neighbors), MQDF (Modified Quadratic Discriminant Function) and SVM-linear in handwriting analysis [36, 57]. Here, we use *one-against-all* SVM, since it works better than the *one-against-one* for handwriting-related tasks [121]. The SVM-RBF hyper-parameters (γ and $\mathscr{C}$) are required to be tuned to control the decision boundary and to avoid over-fitting [129]. Such parameters are chosen from a tuning set for the optimal performance of the classifier. This step is known as "model selection". For this model selection, the traditional *grid-searching* technique is used [241]. Here, k-fold *cross-validation* is employed. The choice of grid-searching range, selection of γ and $\mathscr{C}$, and the choice of k are discussed in *Section* 6.2.2.2.

6.2 Experiments and Discussion

In this section, at first, we discuss the databases employed for the experiments.

6.2.1 Database Employed

For experimental analysis, we required a handwritten database having legibility and aesthetic score. We did not find any database with such ground-truth. Therefore, we generated our own corpus with legibility and ground-truth information. For this, we

gathered some available offline Bengali handwritings. On this handwriting, we requested volunteers to put some subjective score of legibility and aesthetics as discussed in *Section* 6.1.1.

For both legibility and aesthetic subjective scores, we selected the range [0, 10], i.e., $L_1 = 0,\ A_1 = 0,\ L_2 = 10,\ A_2 = 10$ (refer to *Section* 6.1.1). This range was divided into 20 bins, i.e., $n_L = 20,\ n_A = 20$. Therefore, both the legibility and aesthetic analysis tasks can be perceived as 20-class classification problems. We denoted class-1 as lowest score, i.e., poor legibility or poor aesthetics, and class-20 as highest score, i.e., best legibility or best aesthetic opinion.

For a handwritten text, at least 200 human readers provided the subjective legibility and aesthetic scores, and the arithmetic mean opinion score was chosen as the golden standard for each text.

To obtain the legibility score, a human volunteer was shown the cropped handwritten character images arbitrarily, since sequential character showing from a word/text-line may suggest about the writing content due to the semantic knowledge (refer to Figure 6.1). Here, the volunteers were provided the actual character information to remove the false legibility opinion due to erroneous character formation. For example, if a reader is shown only the blue-boxed character of Figure 6.1:(a), and asked to put the subjective score without providing the character information, then (s)he generally gives high legibility score perceiving as the character 'u'. Therefore, if the reader is asked to deliver legibility score after knowing that it is actually 'a', then (s)he can connote the deviation of character formation and be able to provide the proper legibility opinion score for better ground-truthing. For delivering the aesthetic score, the human volunteers were requested to see the full page writing at a distance from 12-18 inches approximately.

For providing the subjective score continuously, the human eyes may be tired after a certain time and the sensitivity of human cognition may deteriorate. To get rid of this, we provided a break to the volunteer after collecting either the legibility scores of 100 characters/ortho-syllables, or aesthetic scores of 50 handwritten pages.

The legibility opinion scores were collected from the native Bengali volunteers who can read and comprehend Bengali writing. The aesthetic scores were collected from volunteers of two separate groups: (a) G_{Ben}: native Bengali persons, (b) G_{nonBen}: non-Bengali persons who cannot read Bengali writing.

The employed Bengali handwritings are as follows.

(i) NewISIdb: HwC [44] (say, D_C): This handwritten character database contains a total of 212,300 Bengali characters of 193 distinct types.

(ii) NewISIdb: HwW [45] (say, D_W): This database contains a total of 118 high-frequency Bengali words written by various writers. Here, the total number of words is 47,200.

(iii) NewISIdb: HwP_Basic [45] (say, D_{PB}): This database contains 300 copies of handwritings of a Bengali paragraph, written by multiple writers. This paragraph contains all the 50 basic characters (11 vowels and 39 consonants) of Bengali script.

(iv) NewISIdb: HwP_Conjunct [45] (say, D_{PC}): 50 copies of a text document, containing most of the conjunct characters are present in this database. This text is big enough containing 595 words. A single copy usually takes 2-3 pages to write this text. Hence, this database contains a total of 136 pages of writing.

(v) CMATERdb1.1.1 [188] (say, D_{CM}): This subpart of the database CMATERdb1 [188], contains 100 Bengali handwriting pages.

(vi) ICDAR-2013 Segmentation [166] (say, D_{DAR}): This dataset contains 50 pages of Bengali handwritten document provided during ICDAR-2013 handwriting segmentation contest [166].

(vii) Struck-out Text [36] (say, D_{ST}): This database comprises 250 Bengali handwritten document images containing some struck-out texts. This dataset details can be found in [36].

The datasets D_C, D_W, D_{PB}, D_{PC}, D_{CM} were used for the legibility analysis, and the datasets D_{PB}, D_{PC}, D_{CM}, D_{DAR}, D_{ST} were employed in the aesthetic analysis. The set D_{C1} contained 50% data of D_C, and D_{C2} contained remaining 50% data of D_C, i.e. $D_{C1} \cup D_{C2} = D_C$. Similarly, $D_{W1} \cup D_{W2} = D_W$, $D_{PB1} \cup D_{PB2} = D_{PB}$, $D_{PC1} \cup D_{PC2} = D_{PC}$, $D_{CM1} \cup D_{CM2} = D_{CM}$, $D_{DAR1} \cup D_{DAR2} = D_{DAR}$, and $D_{ST1} \cup D_{ST2} = D_{ST}$.

6.2.2 Results and Evaluation

In this subsection, we present the experimental results and analyze the performance of our system.

6.2.2.1 Legibility Analysis

From the overall engaged database, we used 50% data for training and the rest for testing. More precisely, the sets D_{C1}, D_{W1}, D_{PB1}, D_{PC1}, D_{CM1} were used for training, and the sets D_{C2}, D_{W2}, D_{PB2}, D_{PC2}, D_{CM2} were employed for testing.

The CNN-extracted features were fed to a bi-directional recurrent neural model for legibility analysis. For training purpose, we employed the stochastic gradient descent

Table 6.1: Legibility analysis

Set	***F-Measure (%)***					
Test / ***Training***	D_{C2}	D_{W2}	D_{PB2}	D_{PC2}	D_{CM2}	$D_T{}'$ *
D_{C1}	**81.54**	79.65	80.24	68.74	75.73	78.16
D_{W1}	71.63	**78.39**	79.03	61.75	73.91	73.29
D_{PB1}	67.74	72.64	**77.84**	57.33	65.20	67.47
D_{PC1}	78.16	79.27	81.48	**77.12**	76.56	79.47
D_{CM1}	74.36	75.41	79.24	60.85	**78.26**	73.28
D_{C1} + D_{W1}	82.34	80.18	80.74	70.48	76.83	79.26
D_{C1} + D_{W1} + D_{PB1}	82.56	80.79	81.66	70.93	77.31	78.75
D_{C1} + D_{W1} + D_{PB1} + D_{PC1}	84.79	84.27	86.83	79.65	81.68	84.43
D_{C1} + D_{W1} + D_{PB1} + D_{PC1} + D_{CM1}	85.16	84.90	87.39	79.72	83.17	**85.74**

* $D_T{}' = D_{C2} + D_{W2} + D_{PB2} + D_{PC2} + D_{CM2}$

with a momentum term of 0.9. The initial learning rate was 10^{-3}, decreased by 10% until the stable validation loss. The training epochs were increased from 1,000 up to 40,000.

The legibility analysis performances, in terms of F-Measure, are shown in Table 6.1. We trained the classifier with the different combination of datasets and also tested on various data samples.

Altogether, executing on the total test-set ($D_T{}'$), we have obtained 85.74% F-Measure, while training has been performed on entire training-set.

6.2.2.2 Aesthetic Analysis

Here also, we used 50% of the overall employed database for training, and the rest 50% for testing. In other words, D_{PB1}, D_{PC1}, D_{CM1}, D_{DAR1}, D_{ST1} were employed as the training set, and the sets D_{PB2}, D_{PC2}, D_{CM2}, D_{DAR2}, D_{ST2} were used for testing.

For aesthetic analysis, the hand-crafted features were fed to the SVM-RBF classifier. The hyper-parameters (γ and $\mathscr{C}$) of the SVM-RBF were tuned using a tuning set. The best performance was obtained for $\gamma = 2^4$ within the range $[2^{-3}, 2^{-2}, \ldots, 2^6]$ and $\mathscr{C} = 2^3$ within the range $[2^{-3}, 2^{-2}, \ldots, 2^7]$. Here, 5-fold cross-validation was used.

Table 6.2 and Table 6.3 present the aesthetic analysis performance (in terms of F-Measure) on subjective opinion scores, collected from G_{Ben} and G_{nonBen}, respectively.

Overall, by executing on the entire test-set ($D_T{}''$), after training by the whole training-set, we obtained 86.69% and 85.33% F-Measure while employing the subjective aesthetic scores of G_{Ben} and G_{nonBen}, respectively.

From Table 6.2 and Table 6.3, it can be observed that the aesthetic analysis perfor-

Table 6.2: Aesthetic analysis by G_{Ben}

Set	***F-Measure (%)***					
Test / ***Training***	D_{PB2}	D_{PC2}	D_{CM2}	D_{DAR2}	D_{ST2}	D_T'' *
D_{PB1}	**84.68**	80.92	82.48	79.67	74.48	80.38
D_{PC1}	81.40	**83.79**	83.02	80.66	75.95	81.82
D_{CM1}	82.76	79.45	**84.42**	80.28	76.36	81.26
D_{DAR1}	82.73	80.04	82.39	**83.27**	75.14	81.33
D_{ST1}	75.49	77.18	76.58	76.76	**80.33**	76.78
D_{PB1} + D_{PC1}	85.47	84.68	83.86	81.28	76.17	83.25
D_{PB1} + D_{PC1} + D_{CM1}	86.93	84.74	84.28	81.97	76.38	82.89
D_{PB1} + D_{PC1} + D_{CM1} + D_{DAR1}	87.45	84.89	85.62	84.07	76.55	84.60
D_{PB1} + D_{PC1} + D_{CM1} + D_{DAR1} + D_{ST1}	87.82	86.18	85.91	84.58	81.78	**86.69**

* $D_T'' = D_{PB2} + D_{PC2} + D_{CM2} + D_{DAR2} + D_{ST2}$

Table 6.3: Aesthetic analysis by G_{nonBen}

Set	***F-Measure (%)***					
Test / ***Training***	D_{PB2}	D_{PC2}	D_{CM2}	D_{DAR2}	D_{ST2}	D_T'' *
D_{PB1}	**85.63**	78.55	82.37	78.28	73.30	80.28
D_{PC1}	81.36	**81.92**	81.65	79.34	72.17	79.87
D_{CM1}	83.64	80.26	**85.74**	80.38	74.78	81.61
D_{DAR1}	81.74	79.84	80.93	**82.06**	73.66	79.45
D_{ST1}	74.29	75.96	77.69	75.69	**78.64**	77.44
D_{PB1} + D_{PC1}	85.78	82.73	83.28	80.15	73.56	81.85
D_{PB1} + D_{PC1} + D_{CM1}	86.31	83.34	87.13	80.79	74.85	82.78
D_{PB1} + D_{PC1} + D_{CM1} + D_{DAR1}	86.58	84.58	87.27	82.50	74.92	84.18
D_{PB1} + D_{PC1} + D_{CM1} + D_{DAR1} + D_{ST1}	87.24	85.52	87.83	83.72	80.25	**85.33**

* $D_T'' = D_{PB2} + D_{PC2} + D_{CM2} + D_{DAR2} + D_{ST2}$

mance is almost similar, while the subjective scores are provided by volunteers of G_{Ben} (native Bengali person) and G_{nonBen} (non-Bengali person) groups. In Figure 6.4, we show this graphically while the testing is done on D_T'' after multiple training sessions.

In Figure 6.5, we present the effect of different hand-crafted features for aesthetic analysis employed in an SVM-RBF classifier. Here, the testing is performed on entire test-set (D_T''), while the training is executed on the whole training-set. The x-axis of the plot presented in Figure 6.5 denotes the number of cumulative features, i.e., '1' indicates feature f_1, '2' indicates $f_1 + f_2$, '3' indicates $f_1 + f_2 + f_3$, and so on. The performance of the

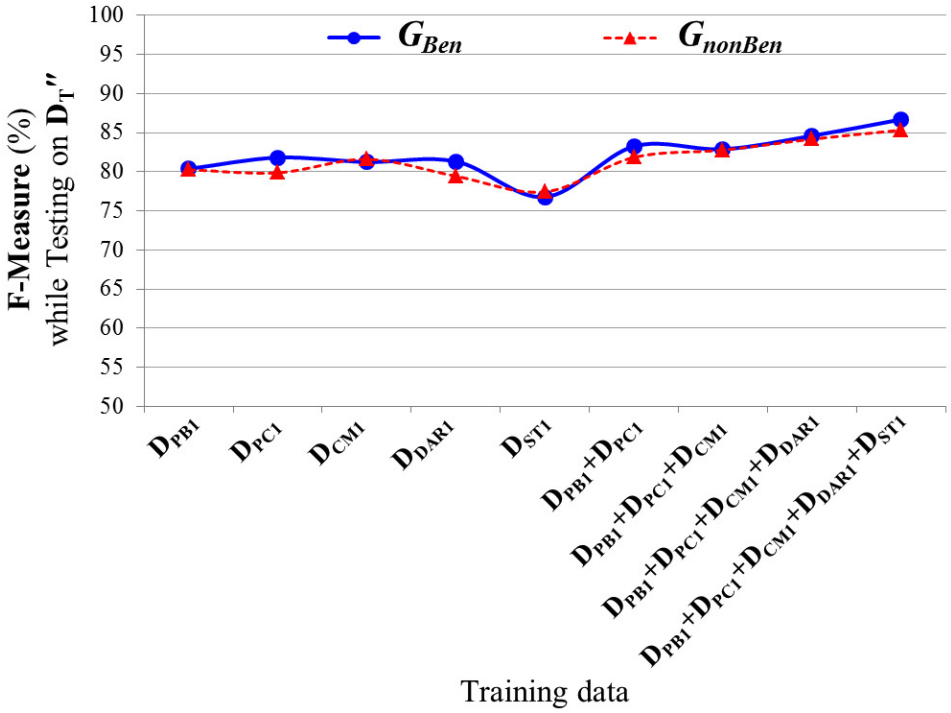


Figure 6.4: Aesthetic analysis with subjective scores of G_{Ben} and G_{nonBen} groups.

aesthetic analyzer correlates positively with increasing of the number of features.

6.2.3 Comparison

Among the works reported in [23], [210] and [234], only that of Majumdar et al. [23] can be compared to some extent. The different motivations of the works of [210] and [234] with different outcomes, restrict us to provide any possible comparison.

Majumdar et al. [23] worked with an in-house dataset of 1200 English handwritten pages and intended to release 275 document pages with ground-truth (unavailable on their project webpage up to 31st December, 2018). For aesthetics ground-truth generation, they employed a total of only 18 persons and each page was annotated by at least 3 individuals using a discrete 5-point Likert scale. For word-level neatness/fine-level assessment, they obtained 55.88% of best accuracy using SURF (Speeded-Up Robust Features). Besides, their system achieved 48.03% best accuracy for page-level neatness/coarse-level assessment employing some statistical features obtained from word segmented image.

We worked with 836 pages of Bengali handwriting for aesthetic analysis. For the ground-truthing, on each page, at least 200 human readers provided the subjective score on a continuous scale. Overall, training on the entire training dataset and testing on

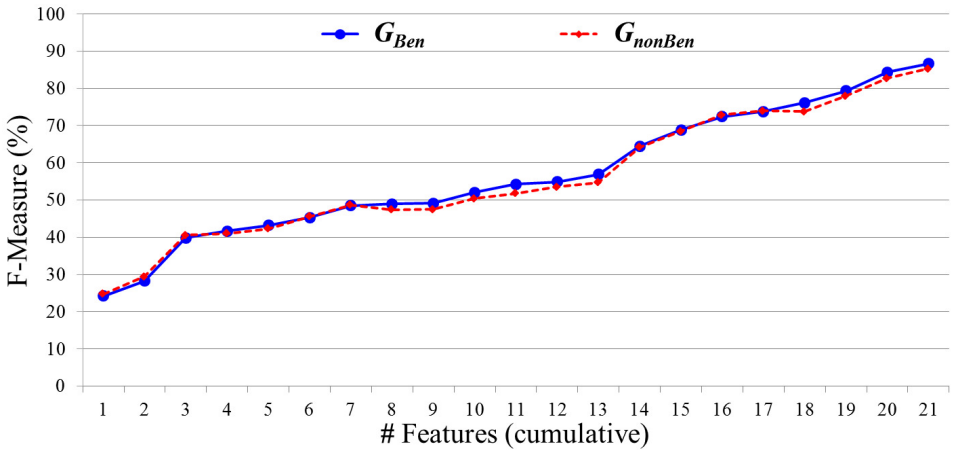


Figure 6.5: The impact of different hand-crafted features for aesthetic analysis, while testing on D_T'' after training by entire training set.

the whole test dataset, we obtained 86.69% and 85.33% F-Measure while employing the subjective aesthetic scores given by groups of native Bengali persons (G_{Ben}) and non-Bengali persons (G_{nonBen}), respectively. We also worked on character-level legibility and obtained 85.74% F-Measure while testing on total test-set after training on the entire training-set.

By searching through online and offline archives to the best of our effort, we did not come across any other related work for performing a comparison.

6.3 Summary

In this chapter, we work on the legibility and aesthetic analysis from Bengali handwritten document image. These tasks are formulated as feature-based classification problems. We employ the auto-derived CNN extracted features and hand-crafted features for legibility and aesthetic analysis, respectively. The auto-derived features are fed into a recurrent neural network and the hand-crafted features are supplied to an SVM model. By gathering some Bengali offline handwriting, we generate a corpus having ground-truth information of legibility and aesthetic subjective scores. Experimenting on this corpus, we obtain an overall 85.74% F-Measure for legibility analysis. The aesthetic analysis provides an overall 86.69% and 85.33% F-Measure while employing the opinion scores of native Bengali (G_{Ben}) and non-Bengali (G_{nonBen}) people, respectively. In the future, we will try to exploit the legibility and aesthetics issues in multi-lingual environments.

CHAPTER 7

DIFFICULTY ANALYSIS FOR WRITING AND READING OF CHARACTER STROKES

Writing is a process of making patterns/graphical symbols to represent the phonology of a spoken language and *Reading* is a process to comprehend the meaning of the writing. For such comprehension, human vision begins at the sensory organ eyes and is processed in the visual cortex of the brain. In the brain, *cognition* plays a vital role in this comprehension. "*Cognition*" refers to "the mental action or process of acquiring knowledge and understanding through thought, experience, and the senses"[1].

It is noted that some human beings find it difficult to properly read or write some characters of a certain script [35]. This problem may be interesting to understand the cognitive perspective of the reading/writing difficulty in comprehension/generation of a handwritten stroke.

In this chapter, we try to investigate the difficulties which may arise in reading and writing. We analyze the subjective opinion score [176] of difficulty collected from several humans who assess the degree of difficulty using their own cognitive system. Providing such subjective rating scores for video quality assessment is quite common in the video

This work is partially published as:
C. Adak, B. B. Chaudhuri, M. Blumenstein, "Cognitive Analysis for Reading and Writing of Bengali Conjuncts", Proc. 31st Int. Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8-13 July, 2018.

[1]Oxford Dictionary, Online: `https://en.oxforddictionaries.com/definition/cognition`, retrieved on 31st Dec. 2018.

broadcasting domain [176]. Here, we aim to impart to a machine a human-like insight to make an artificial cognitive system that understands the complications appearing in reading/writing. Moreover, such an analysis can be used in the pre-processing module of an OCR (Optical Character Recognition) engine for sorting the texts with respect to the reading/writing difficulties [210]. Then the complex one may be sent to a different branch of the OCR system, which boosts the accuracy.

In the *Educational research* [151] domain, the philosophical aspects of reading and writing have been studied for more than 100 years [40]. Some *education*-based logical strategies for English learning can be found in [50]. It is also popular in *psychological* studies [34], where the main concern is the mental development process of reading and writing skills. Some small case-based statistical analysis can also be found, e.g., work on Arabic script learning by Taouk and Coltheart [158]. However, a computer-assisted automated approach for such reading/writing analysis is rare. A very preliminary computational approach, mainly discussing the rational aspects of remembering Bengali basic characters, is reported in [35].

This chapter is an early attempt at the automated analysis of cognitive perspectives related to reading/writing complications. We perform our task on an Abugida Indic script, Bengali [37, 178], owing to its complex patterns. Here, we formulate our reading/writing difficulty analysis task as a feature-based classification problem. For our task, we use two separate models: (a) an auto-derived feature-based Inception network, and (b) a hand-crafted feature-based SVM (Support Vector Machine). We also employ several human individuals to put some cognitive opinion score on Bengali conjunct characters as groundtruth. We note that a large population faces problems in reading/writing of some Bengali conjunct characters rather than basic characters [35]. Therefore, we analyze our method on Bengali conjuncts only. In *Section* 7.1, we discuss on Bengali conjunct characters. The rest of the chapter is arranged as follows. *Section* 7.2 describes the proposed methodology. After that, the experiments and results are presented in *Section* 7.3. Finally, *Section* 7.4 summarizes this chapter.

7.1 Briefings on Bengali Script

The Bengali (endonym, *Bangla*) language is used by more than 250 million native speakers located mostly in south-east Asia. It is the national and foremost official language of Bangladesh, and one of the official languages of India.

Bengali script, evolved from Brahmi script, is about 1000 years old and its present

form came into being in 17th century AD [178]. In modern Bengali script, more than 300 characters are present, among which 50 characters are basic [37]. More than one basic character (up to four characters) can be joined together to create a conjunct character. The number of such conjunct characters is more than 250. Besides, some conjunct characters are written in several shapes. This makes the conjunct character set even larger and more complex in structure. Basically, two types of conjunct characters are formed by typologists. They are called transparent and non-transparent [35]. In transparent conjuncts, the basic character shape remains nearly the same. However, in the non-transparent conjuncts, the basic character shape may change drastically. Some examples of printed and handwritten Bengali conjuncts are shown in Figure 7.1 and Figure 7.2, respectively.

হ্ + ম	ঞ্ + চ	ক্ + ত্ + ব
হ্ম	ঞ্চ	ক্ত্ব
হ্মা	ঞ্ছ	ক্ত্ত্ব

Figure 7.1: Printed Bengali conjunct characters: 1st row shows conjoining basic characters, 2nd row depicts transparent conjuncts (*Bangla Akademi* font), 3rd row represents non-transparent conjuncts (*AdorshoLipi* font). Each column is comprised of the same conjunct.

Moreover, handwritten Bengali conjuncts introduce additional patterns due to the intensive variation of writing styles and cursiveness [44, 45].

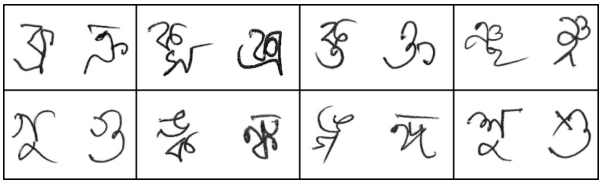


Figure 7.2: Handwritten Bengali conjunct characters: Each box contains a pair of the same conjunct, written in two different ways.

More detailed descriptions of Bengali language and script characteristics can be found in [35, 37, 44, 178].

7.2 Proposed Method

In this section, at first, we formulate the problem and then discuss our proposed method.

7.2.1 Problem Formulation

The sense of reading and writing difficulties depends on human perception and here we attempt to teach the machine this cognitive perceptivity in a quantitative way. Here, the problem formulation tactic is quite similar to the scheme of *Section* 6.1.1 of *Chapter* 6.

Some conjunct characters both in isolated form and word-within form are shown to human volunteers and asked to read/write them. Then the volunteers are requested to put a subjective score of reading-difficulty within a continuous range of $[r_1, r_2]$; $r_1, r_2 \in \mathbb{R}$. Similarly, for writing-complication analysis, the human writer is asked to provide a score in a continuous range $[w_1, w_2]$; $w_1, w_2 \in \mathbb{R}$. During data collection, the subjective scores are marked in a continuous range, since primarily we avoid restricting the scores to a fixed discrete opinion scale, e.g., *Likert scale* [185]. For each conjunct character, several individuals provide their opinion scores. The arithmetic mean of all these scores is calculated and labeled with each character as its *mean opinion score*.

After data collection, the reading-complication score range $[r_1, r_2]$ is partitioned into n_r bins. Therefore, all the *reading mean opinion score*s of the entire character set fall into these n_r bins. In other words, all the conjunct characters are categorized into n_r classes with respect to the reading-complication score. Similarly, the conjunct characters are tagged with *writing mean opinion score*s distributed into n_w bins of score range $[w_1, w_2]$.

Thus, we map the cognitive complication analysis of reading and writing into classification problems. Now, the task is to label a given conjunct into n_r classes of reading and n_w classes of writing opinion score. Such a categorization of opinion scores is quite a well-known topic in the video quality assessment domain [176] and relatively new in the field of document image analysis [49].

The reading/writing difficulty analysis task is performed using two separate methods: an auto-derived feature-based Inception network and a hand-crafted feature-based SVM. These methods are discussed as follows.

7.2.2 Auto-derived Feature-based Inception Network

In this subsection, we discuss the auto-derived features employed for classification of conjunct characters with respect to the reading/writing effort opinion score.

A standard CNN (Convolutional Neural Network) architecture [254] can be considered as a conjoint of two parts: *front* and *rear*. The *front* part is the convolutional part, considered as an automated feature extractor, which takes an image input and produces the feature vector. The *rear* part is usually an MLP (Multi-Layer Perceptron) that is

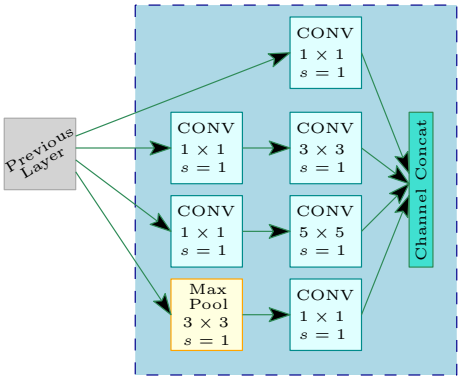


Figure 7.3: Inception Module.

used as a classifier. The auto-extracted features are fed to this MLP for obtaining the classified output through the stacked fully connected layers.

For our reading/writing analysis task, we adapt the deep neural architecture of [63], where the concept of an *"inception module"* is reported. In Figure 7.3, we show the inception module that is employed in our work. Here, "CONV" denotes the convolutional layer, "Max Pool" represents the max-pooling layer, s denotes the stride value. The filter size and stride value used in our experiment are provided in Figure 7.3.

We adapt the idea of an inception module, and work with a relatively smaller version of [63] due to a fewer class involvement in our problem and to handle memory complexity. Our adapted version of the Inception Network is presented in Figure 7.4. This architecture takes a fixed-sized character image input (i/p). We feed a character patch of size 116×116 to the network. In the front part of the Inception network, the first convolutional layer (CONV) contains 64 feature maps of size 112×112. Each of these

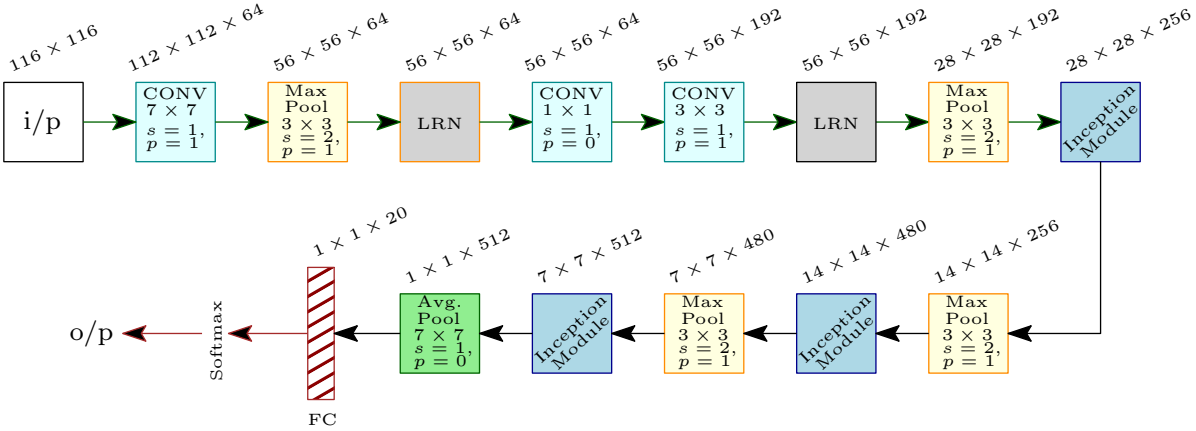


Figure 7.4: Our adapted Inception Network model.

feature maps is linked with a 7×7 neighborhood of the i/p image. Here, stride $s = 1$ and padding $p = 1$ are used. The following max-pooling (Max Pool) layer, LRN (Local Response Normalization), CONV layers are shown in Figure 7.4 with the corresponding employed filter size, the number of feature maps, map size, stride, padding, etc. The actual inner-view of an inception module of Figure 7.4 is shown in Figure 7.3. In total, we use three inception modules here. We use ReLU (Rectified Linear Unit) [242] as the activation function for all the convolutions including those inside the inception module.

After the last inception module, the rear part of the Inception network contains an average pooling ("Avg. Pool") layer which obeys the technique of [154]. The following layer is fully connected (FC) with a *dropout* of 40% [164]. Finally, *softmax activation* is used to predict the class. The output size of each layer is shown in Figure 7.4.

7.2.3 Hand-crafted Feature-based SVM

In this subsection, we discuss the hand-crafted features employed for categorization of characters with respect to the reading/writing effort opinion score. We use an SVM for this categorization.

7.2.3.1 Feature Extraction

Here, we design some hand-crafted features of a character to analyze the reading/writing difficulties. The extracted features are discussed below.

7.2.3.1.1 Structural Keypoint : A character stroke contains multiple structural keypoints [44], such as *end*, *junction*, *curvature* and *cusp* points [99]. The *end* point denotes the start/stop point of a stroke. The *junction* point is actually the intersection position of multiple strokes or the furcation point of the same stroke. The *curvature* point represents the changing orientation spot of a stroke. The *cusp* point on a stroke is a point where the direction of the stroke changes sharply with the non-existence of a derivative. These keypoints may influence the reading/writing of a character.

Here, we count the end points, junction points, curvature points and cusp points on a character stroke, and use them as feature f_1, f_2, f_3, and f_4, respectively.

7.2.3.1.2 Loop : Some Bengali characters contain loop/hole/delta-like structures [46]. Such structures may be useful for analyzing complications in reading/writing. The count of holes is used as feature f_5.

Let, LP be the set of object pixels ($p \in LP$) in a loop region, whereas we assume LC to be the set of pixels on the loop region boundary. The numbers of object pixels in LP and LC are denoted by $n(LP)$ and $n(LC)$, respectively. The center of gravity (CG) inside the loop region is ($CG.x$, $CG.y$), given by:

$$CG.x = \frac{1}{n(LP)} \sum_{p \in LP} p.x \; ; \quad CG.y = \frac{1}{n(LP)} \sum_{p \in LP} p.y \tag{7.1}$$

We calculate the standard deviation (σ_L) of the loop from not being an ideal circle [46] and use it as feature f_6 to fit in our task.

$$f_6 \equiv \sigma_L = \sqrt{\frac{1}{n(LC)} \sum_{p \in LC} (X_p - \mu_L)^2} \tag{7.2}$$

where, $X_p = \sqrt{(p.x - CG.x)^2 + (p.y - CG.y)^2}$ and $\mu_L = \frac{1}{n(LC)} \sum_{p \in LC} X_p$.

We measure another loop-feature (f_7) reliant on the *roundness* ($\mathscr{R}$) of an object [46], as follows.

$$f_7 \equiv \mathscr{R} = \frac{4\pi.n(LP)}{(\sum_{p \in LC} X_p)^2} \tag{7.3}$$

7.2.3.1.3 Isolated Component : A character may contain some isolated components. Such components are generated due to pen up/down movements, which may obstruct effortless writing/reading. We count these components and use as feature f_8.

7.2.3.1.4 Concavity : The character cavity region can be perceived as a "water reservoir" in 2-Dimensions, where "water" can be "poured" from one side of the character shape [229]. We consider four sides, i.e., top, bottom, left, and right, from where "water" can fill the character's concavity or "water reservoir" until a spill-over occurs (refer to Figure 7.5:(a)). Such a character concavity may effect the process of graceful reading/writing. Therefore, we count the number of the top, bottom, left, and right reservoirs and use these as features f_9, f_{10}, f_{11}, and f_{12}, respectively.

We also calculate the *area* of a reservoir that signifies its capacity or concavity. The total area of top reservoirs is computed as A_T. Similarly, the total area of the bottom, left, and right reservoirs are A_B, A_L, and A_R, respectively. The top-bottom reservoir area difference (A_{TB}) is calculated and employed as feature f_{13}. Likewise, the left-right reservoir area difference (A_{LR}) is computed to use as feature f_{14}.

$$f_{13} \equiv A_{TB} = \frac{|A_T - A_B|}{A_T + A_B} \; ; \qquad f_{14} \equiv A_{LR} = \frac{|A_L - A_R|}{A_L + A_R} \tag{7.4}$$

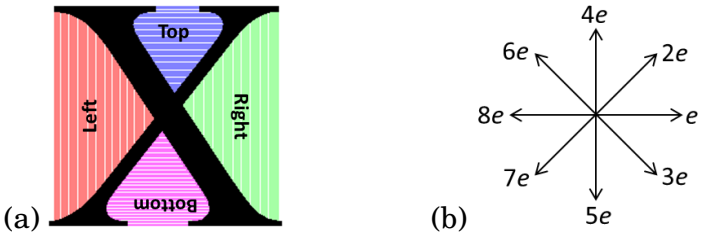


Figure 7.5: (a) Concave water reservoirs: Top, Bottom, Left, and Right reservoirs are shown by the color blue, magenta, red, and green, respectively. (b) Effort measure in 8-directions from a central pixel.

7.2.3.1.5 Straight Line : A straight line in a character shape is formed by combining collinear object pixels. The presence of straight lines in a character may influence how easy it is to read/write. We count the horizontal (—), vertical (|), front-diagonal (/), and back-diagonal (\ ) straight lines, and use these counts as features f_{15}, f_{16}, f_{17}, and f_{18}, respectively.

7.2.3.1.6 Loci : We use a pseudo-structural descriptor called *Loci* [97], which generate an 8-length ternary code, named *Locus Number* (*LN*) [65] by the crossing counts on strokes in 8-directions from the character's *CG* (*Center of Gravity*). We convert the 8-bit ternary number (base = 3) to its corresponding decimal (base = 10).

$$LN(CG) = \sum_{k=0}^{K-1} n.3^k \tag{7.5}$$

where, n is the number of intersections and $n \in 0,1,2$. Here, $K = 8$, since we inspect 8 directions from the CG. This $LN(CG)$ is employed as feature f_{19} in our task. We study this feature due to its obstruction capturing ability, traveling from the character's CG towards the periphery.

7.2.3.1.7 Effort-measure : The *effort* of scribbling a character stroke pattern may be considered as the summation of efforts from one pixel to the next. Let, e be the effort-measure if the stroke is drawn from the one left pixel to the next right pixel through the horizontal axis. From a central pixel, the 8-directional effort measure, adjusted empirically, is shown in Figure 7.5:(b).

Adding effort-measures (e) of all object pixels in a character, let the sum be $E \times e$, where E is a natural number. The *effort-measure* of a character is $(E/N)e$, which undertakes normalization by N. This N is a scale insensitive quantity, calculated as $N = N_{BB} \times E_{BB}$, where N_{BB} is the total number of black pixels in the character bounding

box (BB) and E_{BB} is the elongation of the BB; $E_{BB} = max(H_{BB}, W_{BB})/min(H_{BB}, W_{BB})$, where, H_{BB} and W_{BB} represent the height and width of the BB. The effort measure $(E/N)e$ can be used as a feature having an impact on reading/writing analysis. Let us assume e to be 1, and compute the feature $f_{20} = (E/N)$.

In this manner, we obtain 20 single-valued features from a Bengali conjunct character.

7.2.3.2 SVM Classifier

The feature vector of size 20 is fed to an SVM classifier to categorize a character corresponding to its reading/writing difficulty opinion score. The SVM usually works well for multi-class classification in a wide range of pattern recognition applications [255]. Concerning SVM multiclass classification for handwriting-related tasks, the *one-against-all* strategy works better than the *one-against-one* [121]. It can also be noted from [36, 57] that the SVM with an RBF (Radial Basis Function) kernel [241] works well compared to some other classifiers such as kNN (k-Nearest Neighbors), MLP (Multi-Layer Perceptron), MQDF (Modified Quadratic Discriminant Function), and SVM-linear while dealing with Bengali characters. Hence, we use one-against-all SVM-RBF for our task.

The SVM-RBF hyper-parameters (γ and $\mathscr{C}$) are essential to be tuned to regulate the decision boundary and to avoid over-fitting [129]. For the optimal performance of the classifier, the hyper-parameters are selected from a tuning set, called the *model selection* step. We use the traditional *grid-searching* technique for this model selection [241]. Here, k-fold *cross-validation* is used.

The best performance is obtained for $\gamma = 2^4$ within the range $[2^{-3}, 2^{-2}, \ldots, 2^6]$ and $\mathscr{C} = 2^3$ within the range $[2^{-3}, 2^{-2}, \ldots, 2^7]$. Here, we use 5-fold cross-validation.

7.3 Experiments and Discussion

In this section, at first, we discuss the employed database for experimentation.

7.3.1 Database Employed

For the experimentation, we required a database having a subjective score of difficulties arising in character reading/writing. We did not find any standard and publicly available databases with such ground-truth. Therefore, we generated our own corpus containing subjective opinion ground-truth scores of complications arising in Bengali conjunct reading/writing. For this, we gathered Bengali conjunct characters, on which

we asked volunteers to put some subjective scores while reading and writing separately, as discussed in *Section* 7.2.1.

For both the reading and writing score, we selected the range [0, 10], i.e., $r_1 = 0$, $w_1 = 0$, $r_2 = 10$, $w_2 = 10$ (refer to *Section* 7.2.1). This range was partitioned into 20 bins, i.e., $n_r = 20$, $n_w = 20$. Therefore, both the reading and writing difficulty analysis tasks can be perceived as 20-class classification problems. The idea behind such fixing of the class number was adapted from [176]. The class-1 was represented by the lowest score, i.e., most easy in reading/writing, and class-20 as the highest score, i.e., most difficult in reading/writing.

For each Bengali conjunct character, at least 100 human volunteers provided subjective reading and writing difficulty opinion scores, and the arithmetic mean opinion score was selected as the gold standard for each character.

To obtain the reading opinion score, a volunteer was shown the cropped Bengali conjunct characters arbitrarily, since sequential characters shown from a text-line/word may hint at the actual content due to the semantic knowledge. For delivering the reading score, the human readers were invited to see the character image at a distance of 12-18 inches approximately.

The writing opinion score was also collected at the same time while asking to copy-/write the conjunct character in an offline mode. All volunteers were requested to use a general 0.5 mm ball-point smooth black-inked pen of a particular brand/model to write on a 75 *GSM* (g/m^2) blank white page placed on a common good writing surface.

A continuous subjective scoring may exhaust human eyes and may deteriorate the sensitivity of the human cognitive system. To alleviate this, the volunteer was given a 5 to 10 minutes break after providing reading and writing scores of 50 characters.

The reading/writing scores were collected from native Bengali persons who were able to read and comprehend Bengali writing. The volunteers were of both genders in the age group of 14 to 66 years, having various academic backgrounds and passed at least school grade VIII. The volunteers were generally healthy human-beings without having any known serious illnesses, e.g., Dyslexia [34], Parkinson's disease, etc., which may obstruct reading/writing skill. All the scores were collected during the healthy, awakened state of the volunteers.

For subjective opinion score collection, we employed two commonly used Bengali printed fonts and three databases of Bengali contemporary handwriting; those are discussed as follows.

(*i*) *Bangla Akademi* (say, D_{BA}): This printed font has been developed by *Paschimbanga*

Bangla Akademi, India. In this font, most of the Bengali conjunct characters are made simple for being in the transparent form [35] (refer to Figure 7.1). We typed 220 usual conjunct characters using this font to generate database D_{BA}.

(*ii*) *AdorshoLipi* (say, D_{AL}): This printed Bengali font contains traditional non-transparent conjunct characters [35] (refer to Figure 7.1). A total of 220 conjunct characters, similar to D_{BA}, were typed in this font to generate database D_{AL}.

(*iii*) *NewISIdb: HwC* [44] (say, D_C): A total of 133 distinct types of Bengali conjunct characters written by 100 writers are present in this database.

(*iv*) *NewISIdb: HwW* [45] (say, D_W): In this database, 111 words containing Bengali conjuncts, written by 100 writers are present.

(*v*) *NewISIdb: HwP_Conjunct* [45] (say, D_{PC}): This database contains 50 handwritten copies of a text, contributed by 50 writers. Each copy contains 587 words containing Bengali conjunct characters.

The conjunct characters of the databases D_C, D_W, D_{PC} were separated if they belonged to a word. This separation was performed using a semi-automatic approach, since such conjuncts were in focus. A volunteer was shown at least 30 individual handwritten samples of each handwritten conjunct.

7.3.2 Results and Evaluation

In this subsection, the experimental results are presented to analyze the performance of our system.

7.3.2.1 Auto-derived Feature-based Analysis

We conducted our experiment separately on the previously mentioned five databases D_{BA}, D_{AL}, D_C, D_W, and D_{PC} (refer to *Section* 7.3.1). For all the databases, the training, validation and test sets were partitioned in the ratio of 2:1:1. These training, validation and test sets contained unique conjunct characters with no overlap.

The performance of auto-derived feature-based analysis employing an Inception network on the test set, with respect to the F-Measure (FM), is presented in Table 7.1.

We obtained the highest performance on database D_{BA} and the lowest performance on D_{PC}. We also tested on the overall printed database (D_{BA} + D_{AL}) and overall handwritten database (D_C + D_W + D_{PC}), separately. The performance on the total printed databases was better than the total handwritten database. On the total printed (handwritten) database, we obtained 86.58% (77.23%) and 81.65% (75.84%) FM in reading and writing

Table 7.1: Auto-derived feature-based analysis

Database	***F-Measure (%)***	
	Reading	*Writing*
D_{BA}	92.85	87.47
D_{AL}	84.38	80.92
D_{C}	86.67	83.33
D_{W}	88.18	85.94
D_{PC}	76.02	73.39
D_{BA} + D_{AL}	86.58	81.65
D_{C} + D_{W} + D_{PC}	77.23	75.84

analysis, respectively. Overall, the reading analysis performance was better than the writing analysis.

7.3.2.2 Hand-crafted Features with SVM-based Analysis

In this experiment also, we used the training, validation and test set setup of the auto-derived feature-based analysis (refer to *Section* 7.3.2.1).

The performance analysis employing the hand-crafted feature-based SVM on the test set is shown in Table 7.2.

Table 7.2: Hand-crafted feature-based analysis

Database	***F-Measure (%)***	
	Reading	*Writing*
D_{BA}	86.28	82.94
D_{AL}	78.65	75.54
D_{C}	79.78	77.83
D_{W}	82.40	80.28
D_{PC}	69.27	64.65
D_{BA} + D_{AL}	81.19	77.04
D_{C} + D_{W} + D_{PC}	72.62	69.86

Here also, the highest performance was obtained on D_{BA}, and the lowest was on D_{PC}. The writing analysis performance was worse than the performance for the reading. On the total printed database, we obtained 81.19% and 77.04% FM for reading and writing analysis, respectively. For the total handwritten database, these reading and writing performances were 72.62% and 69.86% FM, respectively.

In Figure 7.6, we show the overall barcode representation of reading and writing difficulty analysis.

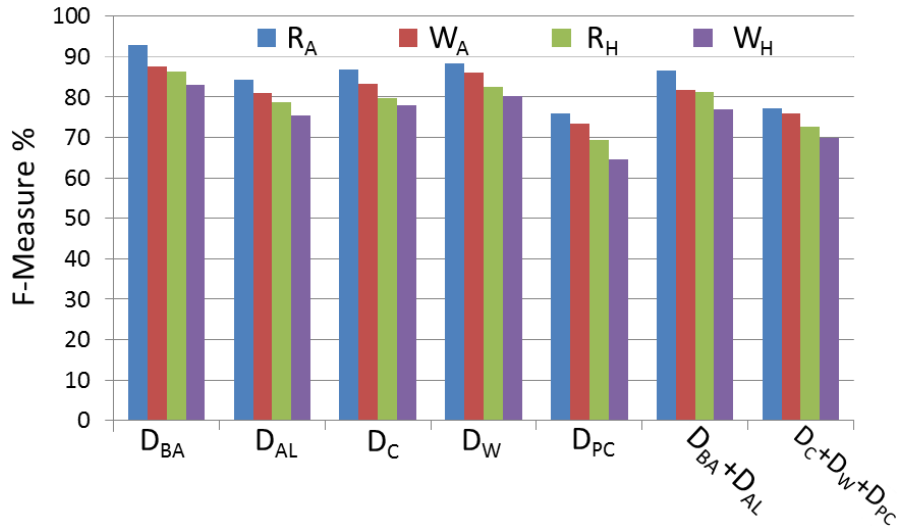


Figure 7.6: Barcode representation of reading/writing difficulty analysis: R_A and W_A denote Reading and Writing analysis using Auto-derived features, R_H and W_H represent Reading and Writing analysis using Hand-crafted features, respectively.

Overall, the auto-derived features worked better than the hand-crafted features. The performance on D_{BA} was better, since it contained mostly the transparent printed conjuncts. The D_{PC} showed poor performance owing to the fact that it contained most of the handwritten complex conjunct characters.

7.3.3 Correlation Testing of Conjunct Character Reading / Writing Difficulty Analysis

Sometimes, the tasks of reading and writing analysis can be thought of as a single problem. Therefore, it was essential to know whether reading and writing complications of Bengali conjuncts correlate. Based on the reading/writing subjective opinion score ranking of conjunct characters, we performed the "*Spearman's rank correlation*" testing [118] and obtained the coefficient value as 0.43. This smaller value signifies lower correlation between reading and writing effort for our task. However, the positive coefficient value denotes that the reading and writing difficulty analysis tasks are not fully opposed.

7.3.4 Comparison

To the best of our searching capacity in both online and offline media, we did not find any papers related to this work for comparison purposes. The work of [35] was very preliminary and mainly a philosophical analysis of remembering Bengali characters, with a lack of proper ground-truthing and experimental setup to be re-performed.

7.4 Summary

In this chapter, we work on Bengali conjunct character reading and writing complication analysis with respect to the subjective cognitive score. The reading and writing analysis tasks are formulated as feature-based classification problems. The auto-derived and hand-crafted features are used for both reading and writing analysis. For experimental analysis, we collect human cognitive scores of difficulties arising in printed and handwritten Bengali conjunct reading/writing, and generate a corpus. By employing auto-derived features on the overall printed (handwritten) corpus, we obtain 86.58% (77.23%) and 81.65% (75.84%) FM in reading and writing analysis, respectively. We also obtain 81.19% (72.62%) and 77.04% (69.86%) FM in reading and writing analysis, respectively, while using the hand-crafted features on the overall printed (handwritten) corpus. Although we work on Bengali conjuncts, our method can be extended to the similar problem of some other scripts. In the future, we will exploit the reading and writing cognitive complexities in multilingual environments.

CHAPTER 8

IDIOSYNCRATIC WRITING STROKE ANALYSIS AND IMPACT ON WRITER IDENTIFICATION

Handwriting is a nexus of graphical marks/symbols where a human writer can be perceived as a "stochastic pattern generator" of such symbols [149]. The structural pattern of such mark/symbol varies with individual due to personal writing style, as discussed earlier in *Chapter* 1. On an individual handwritten sample, it can be noted that some marks/symbols are written in an unconventional way and referred to as "handwriting idiosyncrasy"[1]. According to the Oxford dictionary[2], the word "idiosyncrasy" has been originated from the Greek "*idiosunkrasia*", i.e. *idios* (own, private) + *sun* (with) + *krasis* (mixture), which suggests "a distinctive or peculiar feature or characteristic" of an individual.

In this chapter, the *idiosyncratic handwriting* interprets individual eccentricities in penning a character stroke. In Figure 8.1:(a), a typical idiosyncratic style of writing English character 't' is shown within a red box. Here, the horizontal stroke of 't' is scribbled with a continuity of the vertical stroke. Hence, this 't' apparently looks like cursive lowercase 'f' ('ℓ') and depicts individual idiosyncrasy. Also, Figure 8.1:(b) presents an example of Bengali idiosyncratic handwriting, where the red-boxed character 'ণ' is

This work is partially published as:
C. Adak, B. B. Chaudhuri, M. Blumenstein, "A Study on Idiosyncratic Handwriting with Impact on Writer Identification", Proc. 16th Int. Conf. on Frontiers in Handwriting Recognition (ICFHR), pp. 193-198, Niagara Falls, USA, 5-8 Aug., 2018.

[1]National Records of Scotland, http://www.scottishhandwriting.com, retrieved on 31st Dec. 2018.

[2]https://en.oxforddictionaries.com/definition/idiosyncrasy , retrieved on 31st Dec. 2018.

Figure 8.1: Examples of highly idiosyncratic handwritten character in (a) English and (b) Bengali script, enclosed in red colored boxes.

drawn by noodle-like continuous stroke and creating a queer pattern. Although in this chapter, we mainly consider a complex Indic script Bengali [37, 178], an example in English is presented in Figure 8.1:(a) for mass readership.

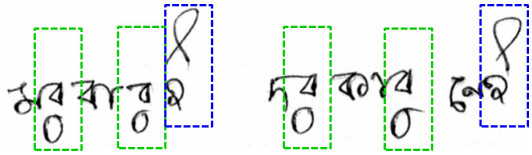


Figure 8.2: Two idiosyncratic characters marked by green and blue dotted boxes.

In Figure 8.2, we present an example of a Bengali text line portion where two idiosyncratic characters (‘র’ and ‘ই’) occur multiple times. Here, a bigger unusual *circular loop* (O) instead of the lower dot component of character ‘র’ and a black ribbon (‘ঁ’) like shape in the upper zone of character ‘ই’ make these highly idiosyncratic.

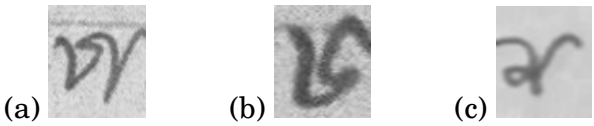


Figure 8.3: Highly idiosyncratic text-patches from *R. N. Tagore*’s manuscript.

In Figure 8.3, we show some examples of idiosyncratic text-patches from the Bengali handwritten old manuscript of Nobel Laureate *R. N. Tagore*. Figure 8.3:(a) depicts the character ‘প’ written in a single stroke without lifting the pen, Figure 8.3:(b) presents the character ‘চ’ looks more likely Bengali numeral ‘৬’, and Figure 8.3:(c) shows character ‘ন’ appears similar to the character ‘ম’.

In the *Educational research* domain [151], very few studies were performed on idiosyncratic writing [116]. On the other hand, in the research field of *Document Image Analysis*, some works mentioned handwriting idiosyncrasy with a motivation for data synthesis [115, 222]. We did not find any other references relevant to our work.

With the assistance of some handwriting experts, we have noted some possible influences of idiosyncratic handwriting generation as follows.

(i) Stroke end / start position: The current character shape may depend on the stroke ending position of the previous character and stroke starting positing of the next character.

(ii) Positional occurrence: A character shape sometimes becomes eccentric due to its start/middle/end position in a word.

(iii) Character stroke component: A single stroke is considered as the scribbling between a pen-down to an immediately next pen-up. Multiple pen-down/ups introduce several character stroke components. Sometimes, to maintain a continuous writing stroke, writer reduces the number of pen-down/ups and may invite idiosyncrasy.

(iv) Cursiveness: The cursive Bengali writing influences mostly to the individual writing styles and eccentricities due to lack of proper handwriting teaching/learning curriculum in school [45]. The students learn cursive writing by their own adaptive skills and sometimes by mimicking the handwritings of some family members, teachers, friends, and scholars. Therefore, sometimes the family/cultural background becomes influential.

(v) Illness: Some illness such as Parkinson's disease, Alzheimer, Dyslexia, Tourette syndrome, etc. may be the cause of idiosyncratic handwriting.

This chapter is an early attempt of computer-based analysis of idiosyncratic handwriting. Here, we formulate our idiosyncrasy analysis task as a feature-based classification problem and use an auto-derived feature-based *Inception network* [63]. We also take assistance from several human handwriting experts to provide some cognitive opinion score on handwritten text-patches as ground-truth.

The study of idiosyncratic handwriting can be used for:

(i) invoking the artificial intelligence into the machine to understand the idiosyncrasy in handwriting,

(ii) sorting the handwritten samples on the basis of idiosyncrasy, which can be utilized in the OCR (Optical Character Recognition) preprocessing stage,

(iii) scrutinizing the forensic and biometric aspects of handwriting,

(iv) investigating distinct penning styles of some famous writers and scholars for the cultural heritage value,

(v) examining early/advanced stage of any physical or neurological disorder which may affect writing styles.

Besides handwriting idiosyncrasy analysis, we further conduct an experiment to see the impact of idiosyncratic handwritten text on writer identification. We note that human handwriting experts emphasize some particular characters/text-patches for unknown

writer inspection [177]. Here, we try to adapt this technique to investigate on highly idiosyncratic text-patches. The writer identification task is perceived as a multiclass classification problem [230], and the auto-derived feature-based SqueezeNet [83] is used here to tackle this problem.

The rest of the chapter is structured as follows. The proposed method is described in *Section* 8.1. Then, *Section* 8.2 provides the experimental results followed by discussion. Finally, the summary is given in *Section* 8.3.

8.1 Proposed Method

A handwritten document is scanned to obtain a digital image and then it undergoes several preprocessing stages. Very small noisy components are removed by a single pass connected component labeling algorithm [87]. Non-textual components such as drawings/doodles are removed using the technique of [42]. Crossed-out/struck-out text, if present in the document image, is also removed by the method of [36]. The characters are segmented using the water reservoir principle-based scheme of [228]. For our task, instead of normalizing the segmented character, we use a fixed size text-patch. The text-patch is selected empirically as a 116×116 neighbor window around the center of gravity of a segmented character. For our task, it is not mandatory that a text-patch should always contain a single properly segmented character. We feed these text-patches for idiosyncrasy analysis followed by the study on writer investigation.

8.1.1 Idiosyncrasy Analysis

In this subsection, at first we discuss our perception of idiosyncrasy analysis task as a machine learning problem, then we describe our approach to tackle this problem.

8.1.1.1 Problem Formulation

The idiosyncratic variation of handwriting is huge in quantity. The sense of idiosyncrasy in writing patterns can be well-examined by human handwriting experts. Here, we attempt to impart the machine with the experts' cognitive experiences instead of some statistical hand-crafted features/characteristics. Here, the problem is formulated by a quite similar scheme of *Section* 6.1.1/*Section* 7.2.1 of *Chapter* 6/*Chapter* 7.

The individual text-patches are shown to the human experts to study the idiosyncrasies. The experts are then requested to provide a subjective idiosyncratic score within

a continuous range of $[I_1, I_2]$; $I_1, I_2 \in \mathbb{R}$. The subjective scores are marked in a continuous range to avoid the restriction of scoring on a fixed discrete opinion scale, e.g., *Likert scale* [185]. For each text-patch, multiple individuals provide their opinion scores. The arithmetic mean of all these scores is calculated and labeled with each text-patch as its *mean opinion score*.

After the data collection, the idiosyncratic score range $[I_1, I_2]$ is partitioned into n_I number of bins of equal span. Therefore, all the *idiosyncratic mean opinion score*s of entire text-patch set fall into these n_I bins. In other words, all the text-patches are categorized into n_I classes with respect to the idiosyncratic score. Thus, we map the idiosyncrasy analysis job into a classification problem. Now, the task is to label a given text-patch into n_I classes of idiosyncratic opinion score. Such categorization of opinion scores is a well-known topic in the field of video quality assessment [176].

The idiosyncrasy analysis task is performed by an auto-derived feature-based Inception network, discussed in the following subsection.

8.1.1.2 Idiosyncratic Text Identification

The idiosyncratic text-patches are marked or classified with respect to the idiosyncratic opinion scores. Some auto-derived features are employed for this purpose and discussed in this subsection.

The standard CNN architecture [254] can be deemed as a knot of two parts: *front* and *rear*, as discussed earlier in *Section* 7.2.2 of *Chapter* 7. The *front* part is the convolutional part regarded as an automated feature extractor, which takes an image input and produces the feature vector. The *rear* part is typically an MLP (Multi-Layer Perceptron) used as a classifier. The auto-extracted features are fed to the MLP for obtaining the classified output through the stacked fully connected layers.

For our idiosyncratic text analysis, we adapt the deep neural architecture of [63], which has reported the concept of *inception module*. In Figure 8.4, we present the inception module (IM) used in this chapter. Here, the "CONV" represents convolutional layer, "Max Pool" denotes max-pooling layer, s symbolizes the stride value. The employed filter size and stride value are provided in Figure 8.4.

We adapt the strategy of inception module and work with a comparatively smaller version of [63] due to lesser class classification and reduced memory complexity. Our adapted version of the *Inception Network* is shown in Figure 8.5. This architecture takes a fixed-sized image input (i/p). We feed the text-patch image of size 116×116. In the front part of the Inception network, the initial convolutional layer (CONV) contains 64 feature

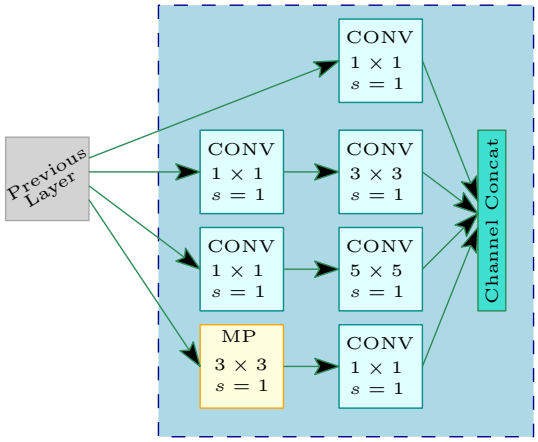


Figure 8.4: Inception Module (IM).

maps of size 112×112. Each of these feature maps is connected with a 7×7 neighborhood of the i/p image. Here, stride (s) = 1 and padding (p) = 1. The following max-pooling (MP) layer, LRN (Local Response Normalization), CONV layers are displayed in Figure 8.5, with the corresponding employed filter size, the number of feature maps, map size, stride, padding, etc. The actual inner-view of inception module (IM) of Figure 8.5 is presented in Figure 8.4. In total, here we employ three inception modules. We apply ReLU (Rectified Linear Unit) [242] as an activation function for all the convolutions including those inside the inception module.

After the last inception module, the rear part of the inception network contains an average pooling (Avg. Pool) layer which obeys the technique of [154]. The following layer is fully connected (FC) with a dropout of 40% [164]. Thereafter, the softmax activation

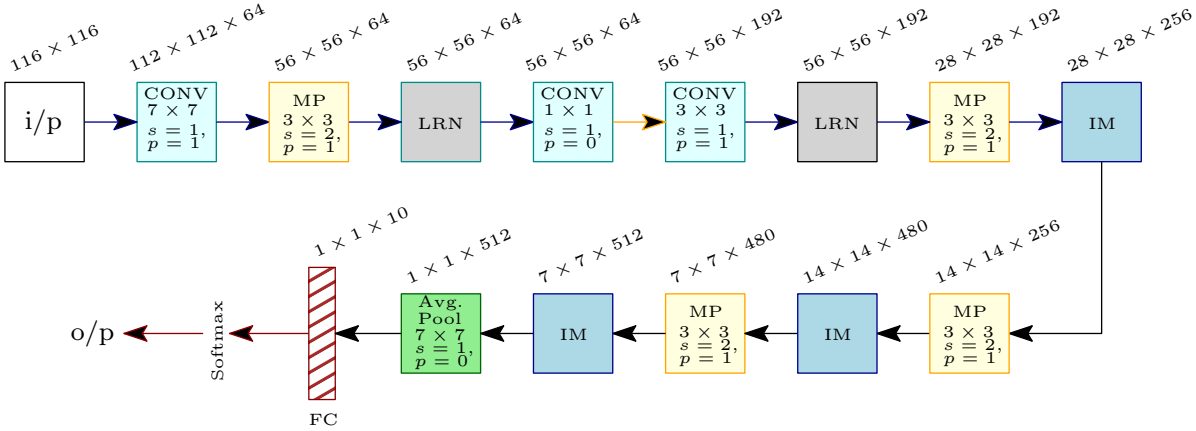


Figure 8.5: Our adapted Inception Network.

is used for predicting the class output (o/p). The output size of each layer is shown in Figure 8.5.

8.1.2 Writer Identification

Writer identification is a task of allocating a writer-id on a questioned handwritten sample. In other words, writer identification is basically a multiclass classification problem to detect a writer-class among multiple classes [230]. Here also, we use an auto-derived feature-based classification strategy. For writer identification, we employ the SqueezeNet [83] architecture, since this architecture provides good accuracy for general image processing tasks with lesser parameter compared to the AlexNet [20].

In SqueezeNet, the key innovation is the *fire module* ($Fire$) [83]. A fire module comprises two layers: *squeeze* and *expand*, as shown in Figure 8.6. The squeeze layer contains $Ns_{1\times1}$ number of 1×1 convolutional filters ($\text{CONV}_{1\times1}$), while the expand layer contains $Ne_{1\times1}$ number of 1×1 convolutional filters and $Ne_{3\times3}$ number of 3×3 convolutional filters ($\text{CONV}_{3\times3}$). Here, Figure 8.6 shows $Ns_{1\times1}$ = 3, $Ne_{1\times1}$ = 4, and $Ne_{3\times3}$ = 4.

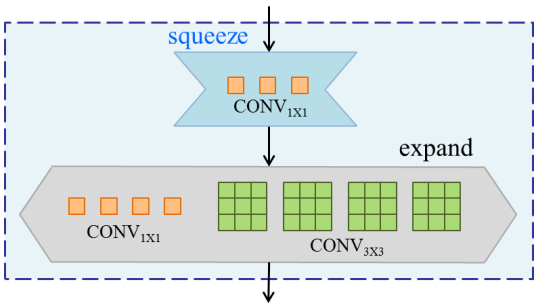


Figure 8.6: Fire module.

For our task, the SqueezeNet takes a text-patch of size 116×116. The first convolutional layer (CONV) comprises of 96 feature maps of size 112×112. Every feature map is connected with a 7×7 neighborhood of the input (i/p) image patch. Here, stride (s) = 1 and padding (p) = 1. The following max-pooling (MP), CONV layers are shown in Figure 8.7 with the corresponding employed filter size, the number of feature maps, map size, stride, and padding measure. Here, we use eight fire modules ($Fire_{i|i=1,2,\ldots,8}$) in total. The counts of ($Ns_{1\times1}$, $Ne_{1\times1}$, $Ne_{3\times3}$) in $Fire_1$, $Fire_2$, ..., $Fire_8$ are (16, 64, 64), (16, 64, 64), (32, 128, 128), (32, 128, 128), (48, 192, 192), (48, 192, 192), (64, 256, 256), (64, 256, 256), respectively. The average pooling (Avg. Pool) layer, after the last convolutional layer, follows the technique of [154]. At last, softmax is used to predict the output class

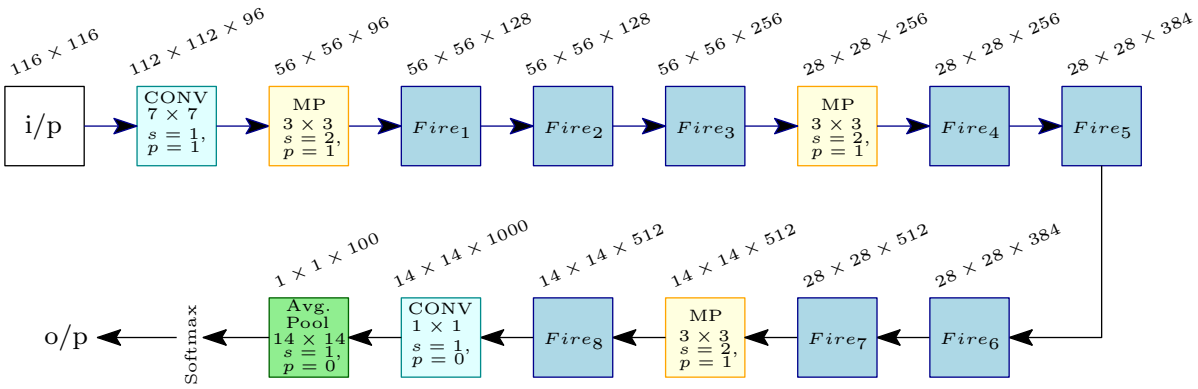


Figure 8.7: Our adapted SqueezeNet architecture.

(o/p). Here, ReLU (Rectified Linear Unit) is used [242] as activation function at the end of both squeeze and expand layers of all fire modules. After the eighth fire module ($Fire_8$), a dropout [164] of 50% ratio is applied.

Finally, the writer is chosen using *majority* rule from a pool of text-patches.

8.2 Experiments and Discussion

In this section, we discuss our employed database and analyze the efficacy of our system through experiments.

8.2.1 Database Employed

For the experimentation, we required a database having a subjective opinion score of handwriting idiosyncrasy. We did not find any database with proper ground-truth. Therefore, we generated our own corpus containing subjective opinion ground-truth score of idiosyncrasy. For this, we gathered the Bengali writings, on which we requested handwriting experts to put some subjective score while examining the idiosyncrasy, as discussed in *Section* 8.1.1.1.

For the idiosyncratic score, we selected the range [0, 10], i.e., I_1=0, I_2=10 (refer to *Section* 8.1.1.1). This range was partitioned into 10 bins, i.e., n_I =10. Therefore, idiosyncrasy analysis tasks can be perceived as 10-class classification problems. The idea behind such fixing the class number was adapted from [176]. The class-1 (ID_1) was represented by the highest score, i.e., most likely to be idiosyncratic writing, and class-10 (ID_{10}) by the lowest score, i.e., least idiosyncratic handwriting.

To obtain the idiosyncratic opinion score, an expert was given cropped Bengali handwritten text-patches irrespective of character class. The expert put the cognitive score by his/her own expertise. For a text-patch, at least 30 human handwriting experts provided the subjective idiosyncratic opinion scores, and the arithmetic mean opinion score was selected as the golden standard for each text-patch.

For subjective opinion score collection, we employed two publicly available databases and one in-house database. The database details are as follows.

i) *ICDAR-2013 segmentation* [166] (say, D_{DAR}): This database comprises 50 Bengali handwritten pages, made available during ICDAR-2013 handwriting segmentation contest [166].

ii) *CMATERdb1.1.1* [188] (say, D_{CM}): This is a subpart of the public database CMATERdb1 [188] and contains 100 Bengali handwritten pages.

iii) *In-house database* (say, D_H): This is an offline database of Bengali handwriting written by 100 writers. Each writer wrote 3 pages to contribute a total of 300 pages.

Here, for D_H generation, a typical 0.5 mm ball-point smooth black inked pen of a particular brand/model was used to write on a 75 GSM (g/m^2) blank white page placed on a common good writing surface. The writers were of both genders with different ages, having various academic backgrounds. These writers were general healthy human-being without having any known serious illness (e.g., Dyslexia, Parkinson's disease, etc.) of obstructing general writing. All the handwritings were performed in the healthy awaken state of the writers.

8.2.2 Results and Evaluation

In this subsection, we discuss the experimental results and our system performances.

8.2.2.1 Idiosyncrasy Analysis Performance

Here, the idiosyncratic analysis task was formulated as a 10-class classification problem where the job was to classify the handwritten patches into idiosyncratic classes (ID_1, ID_2, ..., ID_{10}).

We conducted our experiment of idiosyncratic handwriting analysis separately on the previously mentioned three databases D_{DAR}, D_{CM}, and D_H. Each of these databases with subjective idiosyncratic opinion score was divided into training, validation, and test set in 2:1:1 ratio.

Table 8.1: Idiosyncratic handwriting analysis

Database	***F-Measure (%)***	***Balanced Accuracy (%)***
D_{DAR}	**91.37**	**93.54**
D_{CM}	89.48	91.33
D_{H}	85.96	88.16
D_{DAR} + D_{CM} + D_{H}	89.03	90.89

The performance of our system for idiosyncratic handwritten text-patch identification is presented in Table 8.1. Here we show the overall *F-Measure* (%) of all the classes. Since the idiosyncratic text-patch ground-truths are dependent on experts' cognitive analysis, all classes may not be well balanced. Therefore, our database is quite imbalanced. For classification into such classes, the *balanced accuracy* provides better performance measure [131]. In Table 8.1, we also show the performance of idiosyncrasy analysis in terms of balanced accuracy (%).

On idiosyncrasy analysis, our system obtained the highest performance on database D_{DAR} and lowest performance on D_{H}. For idiosyncratic handwritten text-patch identification, we obtained 91.37%, 89.48%, 85.96% of F-Measure and 93.54%, 91.33%, 88.16% of balanced accuracy on D_{DAR}, D_{CM}, D_{H} databases, respectively. Overall, union of all our employed datasets attained system performance of 89.03% F-Measure and 90.89% balanced accuracy.

8.2.2.2 Writer Identification Performance

In this subsection, we present the experimentations performed for writer identification task to investigate whether there exists any influence of idiosyncratic handwriting on this task.

The databases D_{DAR} and D_{CM} were not suitable for writer identification task, since writer information was missing. The D_{H} contained the writer information and was suitable for our study. To inspect the effect of idiosyncratic handwriting on writer identification, all the obtained text-patches (refer to *Section* 8.1) of D_{H} were subdivided into training, validation, and test set in 2:1:1 ratio.

As previously mentioned in *Section* 8.2.1 and *Section* 8.2.2.1, the handwritten text-patches were classified into 10 idiosyncratic classes (ID_1, ID_2, ..., ID_{10}). The handwritten samples of class ID_1 were highly idiosyncratic whereas ID_{10} contained least idiosyncratic

Table 8.2: Writer identification performance

Idiosyncratic class (ID)	***Accuracy (%)***		
	Top-1	*Top-2*	*Top-5*
ID_1	93.23	93.72	96.57
ID_2	92.85	93.46	95.81
ID_3	91.66	92.18	95.10
ID_4	89.94	90.43	93.76
ID_5	87.38	87.89	91.27
$ID_{1,2}$	94.03	94.76	97.54
$ID_{1,3}$	93.62	93.90	96.79
$ID_{2,3}$	92.98	93.54	95.42
$ID_{1\text{-}3}$	**94.46**	**94.92**	**98.27**
$ID_{1\text{-}4}$	93.71	94.48	97.68
$ID_{1\text{-}5}$	93.28	94.15	96.92
$ID_{1\text{-}10}$	90.43	91.59	94.26

handwritings. We intended to analyze the writer identification performance using only the highest idiosyncratic handwriting (class ID_1) of our database D_H. Besides, we performed the writer identification experiments on different idiosyncratic classes. We also experimented with all possible unary, binary, ternary, quaternary, ..., n-ary combinations of idiosyncratic classes ($ID_{n|n=1,2,\ldots,10}$). Here, in Table 8.2, we present some significant results of our writer identification method on some important idiosyncratic classes.

For writer identification performance analysis, we employed the Top-N criterion where a questioned document's writer belonged to a reduced set of 'N' ($\ll$ total number of writers). Here, we selected Top-1, Top-2 and Top-5 criteria for providing the writer identification performance in terms of *accuracy* (%). In D_H database, the samples of multiple writer classes were quite balanced.

In Table 8.2, $ID_{1,2}$ denotes union of text-patches of the idiosyncratic class-1 (ID_1) and class-2 (ID_2), i.e., $ID_{1,2} \equiv ID_1 \cup ID_2$. Likewise, $ID_{1,3} \equiv ID_1 \cup ID_3$, $ID_{1\text{-}3} \equiv ID_1 \cup ID_2 \cup ID_3$, $ID_{1,2,4} \equiv ID_1 \cup ID_2 \cup ID_4$, etc., and $ID_{1\text{-}10}$ denotes the union of text-patches of all idiosyncratic classes. As a matter of fact, experimenting on $ID_{1\text{-}10}$ is synonymous to the traditional writer identification experiment on the full page of writing.

With reference to Table 8.2, we obtained 93.23% Top-1 writer identification accuracy while employing only the highest idiosyncratic handwriting (class ID_1). On this ID_1, the attained Top-2 and Top-5 accuracies were 93.72% and 96.57%, respectively.

Among binary combinations (e.g., $ID_{1,2}$, $ID_{1,3}$, $ID_{2,3}$, etc.) of the idiosyncratic classes,

highest Top-1 accuracy (94.03%) was obtained on $ID_{1,2}$. Among ternary combinations (e.g., $ID_{1\text{-}3}$, $ID_{2\text{-}4}$, $ID_{1,2,4}$, etc.), highest 94.46% of Top-1 accuracy was obtained on $ID_{1\text{-}3}$. It was the best Top-1 accuracy among all possible n-ary combinations of idiosyncratic classes.

Union of all the idiosyncratic classes ($ID_{1\text{-}10}$) produced 90.43% of Top-1 accuracy. In other words, classical experimentation on all the text-patches attained 90.43% Top-1 writer identification accuracy. However, each of the upper three idiosyncratic classes (ID_1, ID_2, and ID_3) performed better than the traditional approach of considering all text components. Thus, from our experimental results, it can be noted that highly idiosyncratic handwriting can improve the writer identification performance. Here, experimenting on ID_1 and $ID_{1\text{-}3}$, we obtained respective 2.8% and 4.03% additional Top-1 writer identification accuracy when compared to $ID_{1\text{-}10}$. This baseline performance added value to our idiosyncratic analysis.

8.2.3 Comparison

To the best of our knowledge, this work is the earliest attempt of its kind on idiosyncratic handwriting analysis. We did not find any related work for making a comparison.

However, it was interesting to see that highly idiosyncratic handwritings produced up to 4.03% additional Top-1 writer identification accuracy on our experimentations when compared to the same writer identification method without using idiosyncratic analysis (refer to *Section* 8.2.2.2).

8.3 Summary

In this chapter, we scrutinize the idiosyncrasy of individual handwriting and demonstrate its conducive nature for the writer identification task. The idiosyncratic handwriting analysis task is modeled as a machine learning problem with the aid of cognitive score of handwriting experts. Here, an Inception network is employed for idiosyncratic text inspection, and the SqueezeNet is used for writer identification. For both of these tasks, auto-derived deep features are used. Experimentation of idiosyncratic handwriting analysis is performed on two publicly available and one in-house offline Bengali handwritten databases with collected subjective opinion ground-truth. Overall 90.89% balanced accuracy has been obtained for the idiosyncrasy analysis task on these three datasets. The writer identification job has experimented on the in-house database. For unary idiosyn-

cratic classes, we have obtained highest 93.23% Top-1 writer identification accuracy (on ID_1). The best Top-1 accuracy of 94.46% has been obtained while combining three highly unary idiosyncratic classes (ID_1, ID_2, and ID_3). Our experiments suggest that prior analyzing of idiosyncratic handwriting can improve the writer identification performance up to 4.03% of Top-1 accuracy.

Although in this chapter, we have worked on Bengali handwriting, our method can be extended to some other scripts. In the future, we will try to exploit the idiosyncrasy analysis for multilingual handwriting. We also plan to hybridize different neural networks for a comparative study.

Part III

Understanding the Content

CHAPTER 9

CONTENT REVEALING WORD RECOGNITION FROM MANUSCRIPT IMAGES

Automatic character recognition of an optically scanned document is one of the most fascinating research areas of pattern recognition. The literature of the OCR (Optical Character Recognition) is very large and the state-of-the-art OCR engines (e.g., ABBYY FineReader) have achieved reasonably high accuracy for printed documents. However, past OHCR (Optical Handwritten Character Recognition) engines [14, 187] have not fared as well in terms of performance.

On old and degraded documents, the OCR/OHCR engines may not perform well. Therefore, some alternative has been thought of due to the imminent need of document e-archiving in the digital world. At this point, "*keyword spotting*" [238] has come into place, where the task is to locate the instances of a given word in a document. It can play a remarkable role in document indexing and retrieval.

In keyword spotting task, the query word is decided by the user at first, and then detects its existence in a document image. Now, to choose the keywords, reverse engineering may be necessary by peeking into the generation information. In this chapter, we attempt to find some keywords or important words, which may reveal the content of a document. Here, we consider the unstructured handwritten document. For this task,

---

This work is partially published as:
C. Adak, B. B. Chaudhuri, M. Blumenstein, "Named Entity Recognition from Unstructured Handwritten Document Images", Proc. 12th IAPR Int. Workshop on Document Analysis Systems (DAS), pp. 375-380, Santorini, Greece, 11-14 Apr., 2016.

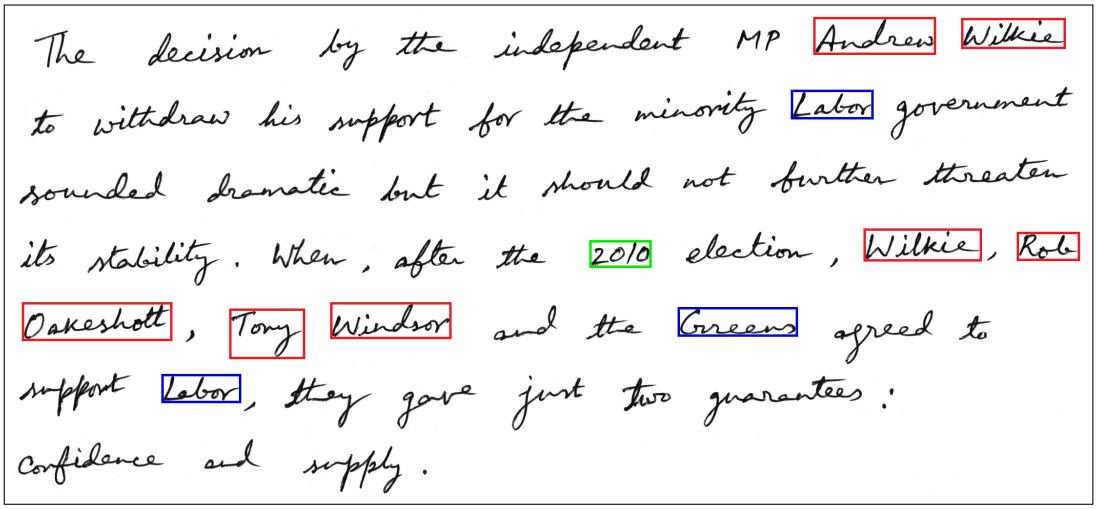


Figure 9.1: Content-revealing named entities of person, organization and year, marked by the red, blue and green color boxes, respectively.

transcribing the textual matter with an OHCR and processing the language by an NLP model (Natural Language Processing) is a straight-forward way, but a costly process due to the aid of OHCR and NLP models. Besides, completely automated transcribing of the old manuscript is yet at risk of underperforming by the state-of-the-art OHCR engines. Therefore, we intend to provide a smaller set of words that can infer the content, and bypass the direct use of OHCR and NLP engines. Thus, instead of machine-reading of the entire document, we reduce the search space and focus on the limited number of content revealing important words. Such words inform about date/time, place, numeric value, currency, person, organization, etc. As a matter of fact, this is mainly name-of-things of living or non-living, and commonly referred to as *Named Entity*. It can also be observed that the "entity names form the main content of a document" [88]. Therefore, in this chapter, we strive to identify handwritten words, which represent named entities and infer the content of the writing. In Figure 9.1, we provide an example of an unstructured handwritten sample, and mark the named entities which help in revealing the content of the writing.

Named Entity Recognition [123] has been a popular topic in the fields of *Natural Language Processing* (NLP) and *Information Retrieval* (IR) for the last two decades. In the NLP domain, a detailed survey on named entity recognition from natural language is reported in [67]. However, work on named entity identification directly from document images is rare. One approach of named entity extraction from a document image was attempted by Zhu et al. [96], who considered semi-structured and unstructured printed

documents of a special type, i.e., "automated expense reimbursement". In their method, a CRF (Conditional Random Field) framework with some rich page layout features was used. They also employed an OCR engine and recognized the named entities with assistance from the OCR outputs.

Without character/word recognition, *Named Entity* (NE)/*Content-revealing Word* (CW) detection from a document image is very difficult because NLP-based knowledge can hardly be used in such a situation. However, such detection is essential where linguistic knowledge cannot be used due to the poor performance of handwritten text recognition engines. In this chapter, we attempt to fill this gap by proposing an approach for NE/CW recognition without performing OCR on the document image. We find some features by analyzing the structural and positional characteristics of named entities and feed them to a BLSTM (Bidirectional Long Short-Term Memory) recurrent neural network to reveal the NE/CW. Our method can process unstructured handwritten offline historical and contemporary English documents. We have not found any published work on NE/CW recognition directly from document images, which is related to the one described here.

In fact, the present problem is completely different from the classical *keyword spotting* [238] problem. In keyword spotting, a template of a keyword is provided to find its matches in a document image. However, in our proposed work, no such template is provided at all. We just use some structural and positional characteristics of an NE/CW for its identification.

The rest of the chapter is organized as follows. *Section 9.1* describes our proposed method in detail. The results and evaluation of the system are provided in *Section 9.2*. Then *Section 9.3* summarizes this chapter.

9.1 Proposed Method

In this section, we elaborate our method of *Named Entity* (NE)/*Content-revealing Word* (CW) recognition. An unstructured document image I is given, and our aim is to identify the NE/CW from this image, which can infer the content of the document.

A workflow of the proposed method is shown in Figure 9.2. After some preprocessing, the document image I is segmented into words $W = \{W_1, W_2, \ldots, W_n\}$, where n denotes the count of words in I. Then, we use a feature-based neural network model to extract a set of named entity/content-revealing word images, i.e., $S_W \in W$. After that, an optional word recognition model is used to produce machine-readable output (S'_W) of NE/CW images (S_W). Both the S_W and S'_W infer the content of I.

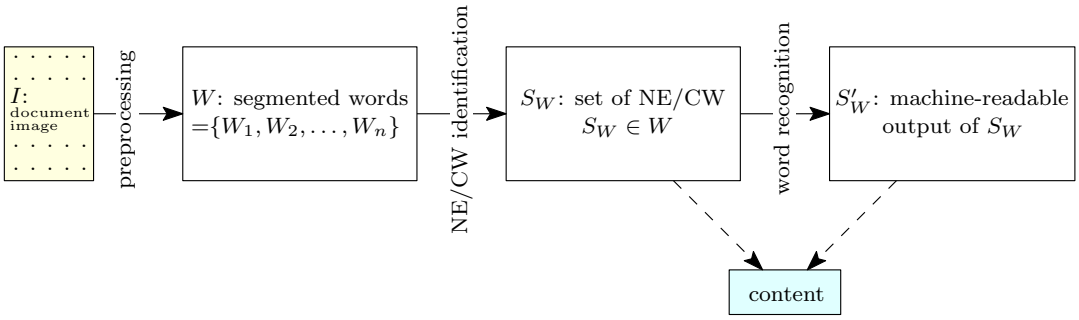


Figure 9.2: Proposed framework.

Now we illustrate each step of our scheme in detail, as follows.

9.1.1 Binarization

A document image $I(r,c)$, where r and c denote row and column index respectively, may contain noise and may be degraded in quality due to aging, insect perforations, bleed-through, discoloration, etc. Therefore, proper binarization is required to analyze the writing strokes. We adapt the binarization technique of Ntirogiannis et al. [134] which is suitable for our purposes for historical documents. A fusion of local and global adaptive binarization with background estimation is used here. The superiority of this method, compared to some other off-the-shelf and recent binarization algorithms, is also shown in [134]. Moreover, we modify this method slightly as per our requirements. In [134], very small components are all rejected as noise, but we retain them and process further to find punctuation marks (in *Section 9.1.4.2*). For modern and good quality documents, Otsu's binarization [163] method works well, and thus, is employed here.

9.1.2 Word Segmentation

Since our aim is to find the NE/CWs which consist of words, we have to segment individual words. We use a 2D Gaussian filter for word segmentation on the image $I(r,c)$. The standard deviation (σ) and block size ($height \times width$) for this filter is chosen automatically by analyzing the connected components in a text-line. In English script, the words are arranged horizontally. Hence, the block size is chosen to be wider in length than height. After blurring, an adaptive threshold is used for binarization and for returning the (word) segmented image.

9.1.3 Slant / Skew / Baseline Correction

In English script, a named entity generally starts with an upper-case character. An upper-case character is normally more elongated in the vertical direction than lower-case ones. This is a useful property and is applied in our approach. However, we may get erroneous results if the characters are slanted. Therefore, slant correction is necessary.

For slant correction of a word, we use the algorithm presented in [74] which fits well for our purpose. This algorithm is based on a vertical projection profile and employs the *Wigner-Ville distribution* [77].

The skew and baseline offsets [231] are also detected and corrected.

9.1.4 Characteristics of Named Entities

Detection of named entity, without any explicit character/word recognition, is challenging because of the lack of linguistic knowledge. The only things available are the patterns of the character strokes; those also vary with individual writers. We analyze some characteristics of named entity from the perspective of image and pattern analysis.

9.1.4.1 Structural Characteristics

A named entity is likely to start with a capital/upper-case letter in English script. It is a useful indicator to identify a named entity. For example, the difference between "*beauty*" (noun of *beautiful*) and "*Beauty*" (may refer to a *person's name*) can be easily identified by the first capital letter in the word image. The height of a capital letter is normally more than that of small/lower-case letters.

To detect a capital letter, we partition a word image into three zones (*upper*, *middle* and *lower*) by two imaginary lines called upper and lower baselines. These baselines are obtained by simple horizontal histogram projections as of the approach, found in [231]. The maximum number of object/ink pixels generally lies in the middle zone, where the horizontal histogram profile reaches the global maximum. This middle zone is actually the main body of a word.

An upper-case character usually covers the upper and middle zones. A capital letter is sometimes wider than a lower-case letter. In the upper projection profile of a word, the maximum height and width are likely to be found near the beginning (left), if the word starts with a capital letter. Some distinct characteristics are also observed in the word projection profiles, if it starts with a capital letter. More details on these aspects are provided in *Section 9.1.5*.

9.1.4.2 Positional Characteristics

A named entity may be positioned at different places of a sentence. It can be broadly categorized in terms of positions: (a) *first* position, i.e. first word of the sentence, (b) *last* position, i.e. last word of the sentence and (c) *middle* position, i.e. anywhere except (a) and (b).

A sentence often ends with punctuation marks such as a *full stop* (.), *question mark* (?), *exclamation mark* (!), etc. which are detected to isolate a sentence from the document image. We note that there is no character in the English alphabet which has a dot-like component in its lower zone. But question marks and exclamation marks have dots in their lower zone. A full stop can be easily detected by finding a dot-like component near the border between the middle and/or lower zone. Such punctuation marks signal the end of a sentence and the beginning of the next sentence. This sentence-end-marking result may be erroneous for degraded and noisy documents, as well as for documents having false end-markers due to individual handwriting habits.

If an NE is positioned in the *middle* or *last* part of a sentence, then the first character of the word is in upper-case. Therefore, by analyzing the position, we can affirm that the *middle* and *last* positioned words are generally NEs, when the first character is a capital letter. Some exceptions are there, which we handle during post-processing.

Now, the difficulty arises with the *first* positional NE, since any *first* positional word usually starts with a capital letter. To know how frequently they occur, we performed positional occurrence analysis of NEs in an English sentence. For this purpose, we took 300 articles (100 newspapers, 100 story books, and 100 online articles). Among these, 10 articles on various topics were chosen from each of the 10 English newspapers having the highest circulation in ten countries. Also, 10 pages from each of the 10 different English story books and novels of distinct writers from various countries were selected. Moreover, 100 online articles on dissimilar topics were chosen from different websites (e.g., various blogs, Wikipedia pages, etc.). At this point, these 300 articles are processed by the *Stanford Named Entity Recognizer* [123], and we find that in an article, at most 16.66% (minimum: 0%) of NEs can reside in the *first* position of a sentence. On average, the *first* positional occurrence of an NE is only 7.01% ($\text{newspaper}_{\text{avg.}}$: 8.33%, $\text{story-book}_{\text{avg.}}$: 5.78%, $\text{online-article}_{\text{avg.}}$: 6.92%). Therefore, the confusion may arise only on the 7.01% of NEs. We deal with such cases later in the post-processing stage.

Obtaining guidance from the *SMART* [95] project, we categorized four special classes of frequently occurring words in the English literature, with respect to structures and

various positions in a sentence. These classes are described below.

(i) Wh-word class: This class contains Wh-question words starting with characters "Wh" and generally positioned at the beginning of a sentence. For example, Why, Who, Whom, What, When, Where, Which, Whose, etc.

(ii) Th-word class: Words starting with "Th"/"th" belong to this class and are usually positioned at the beginning (first) and middle of a sentence. At the first position 'T' is in upper-case, otherwise, it is in lower-case ('t'). For example, The, This, That, These, Those, Thank, etc.

(iii) Small-width word class: This class contains some words with one or two characters, e.g., I, A, An, Am, At, As, Be, We, He, Hi, It, If, Is, Of, On, No, To, In, Dr, Mr, Ms, Mx, Jr, Sr, Lt, Mt, etc.

(iv) All-caps word class: Sometimes, all characters of a word are in capital letters. It is usually in the abbreviation/acronym form of multiple words, Roman numerals, etc., for example, OCR, USD, UAE, XXI, etc.

Some *salutation* or *title* words with a small width, e.g., Mr, Ms, Mx, Md, Dr, Pt, Sk, St, Mrs, Esq, Adv, Sir, Sri, etc., followed by (a *dot* '.' sometimes, and) another word beginning with a capital letter is a sign of NE with high confidence. Also, two or more consecutive words beginning with a capital letter have a higher possibility of being an NE, e.g., Sir Isaac Newton, John F. Kennedy, Victor Dalby Lord Jr., Queen Elizabeth II, etc.

9.1.5 Feature Extraction

The features used in our method of detecting NE/CWs are described below. All features are normalized in the range [0, 1].

9.1.5.1 Object Pixel Distribution

These types of features are based on the distribution of the foreground object/ink pixels throughout the word image.

We consider a vertical sliding window of size $h \times 1$, where 'h' is the height of the word bounding-box. This window is moved from left to right over a word. Then we calculate the *weight* (f_1), *center of gravity* (f_2) and 2^{nd}*-order moment* (f_3) of the window as [231]. Sometimes, f_1 is called a *vertical projection profile*.

$$f_1(c) = \sum_{r=1}^{h} p(r,c) \tag{9.1}$$

$$f_2(c) = \frac{1}{h} \sum_{r=1}^{h} r.p(r,c) \tag{9.2}$$

$$f_3(c) = \frac{1}{h^2} \sum_{r=1}^{h} r^2.p(r,c) \tag{9.3}$$

Here, $p(r,c)$ is the binary image of $I(r,c)$ obtained in *Section 9.1.1* and $p(r,c)$ can take on values 0 or 1, where 1 is the foreground ink pixel.

Additionally, we employ a horizontal sliding window of size $1 \times w$, where 'w' is the width of the word bounding-box, and calculate the features f_4, f_5 and f_6. The f_4 is called as *horizontal projection profile*.

$$f_4(r) = \sum_{c=1}^{w} p(r,c) \tag{9.4}$$

$$f_5(r) = \frac{1}{w} \sum_{c=1}^{w} c.p(r,c) \tag{9.5}$$

$$f_6(r) = \frac{1}{w^2} \sum_{c=1}^{w} c^2.p(r,c) \tag{9.6}$$

9.1.5.2 Profile Features

The word-profile features used in our method are described as follows.

We choose the *upper contour profile* (f_7) [221] of a word, which is the distance from the upper boundary of the bounding-box to the vertically closest object pixel for each column. Similarly, the *lower contour profile* (f_8) is obtained on the lower boundary, and used as a feature.

Also, crossing count of the object (black ink) and non-object (white) pixel for each column is considered as a feature. Crossing count is usually half of the ($black_{to}white + white_{to}black$) transition count. It is called the *vertical black-white transition* (f_9) [231]. Additionally, we compute *horizontal black-white transitions* (f_{10}) of a word.

Furthermore, a *vertical run-length histogram* (f_{11}) and a *horizontal run-length histogram* (f_{12}) are obtained. The run-length is calculated as the sum of consecutive object pixels in the vertical/horizontal direction.

9.1.5.3 Special Weighted Features

In addition to the above twelve features, two weighted features are employed as described below.

On the upper projection profile of a word image, we count the transition of peaks and valleys. Since an NE generally starts with a capital letter, we give some weight on the peak-valley transition count. Let, n_l, n_m and n_r be the count of peak-valley transitions on the left $1/3^{rd}$, middle $1/3^{rd}$ and right $1/3^{rd}$ portion (in x-direction), respectively, of the upper projection profile. This is used as a feature (f_{13}).

$$f_{13} = \mathcal{W}_l.n_l + \mathcal{W}_m.n_m + \mathcal{W}_r.n_r \tag{9.7}$$

Here, $\mathcal{W}_l$, $\mathcal{W}_m$, $\mathcal{W}_r$ are the weight factors, and are chosen empirically as $\mathcal{W}_l = 0.8$, $\mathcal{W}_m = 0.1$, $\mathcal{W}_r = 0.1$.

On the *horizontal projection profile* (f_4), we mark the upper, middle and lower zones, as described in *Section 9.1.4.1*. Since the capital letters normally reside in the upper and middle zone, we apply some weights to this profile feature and obtain f_{14}.

$$f_{14} = \mathcal{W}_t.\sum_{c=1}^{\lfloor \frac{w}{3} \rfloor} p(r,c) + \mathcal{W}_d.\sum_{c>\lfloor \frac{w}{3} \rfloor}^{\lfloor \frac{2w}{3} \rfloor} p(r,c) + \mathcal{W}_b.\sum_{c>\lfloor \frac{2w}{3} \rfloor}^{w} p(r,c) \tag{9.8}$$

Here, 'w' is the width of a word. The weight factors $\mathcal{W}_t = 0.6$, $\mathcal{W}_d = 0.3$, and $\mathcal{W}_b = 0.1$ are chosen empirically.

9.1.6 Classification

In this section, we feed the extracted features to a classifier to detect NE/CWs.

Among a few standard classifiers, we have found that a *Neural Network* yields the highest accuracy for our problem. A detailed discussion on various neural network architectures can be found in [201]. Of the various neural network techniques, we use the BLSTM (Bidirectional Long-Short Term Memory)-based RNN (Recurrent Neural Network) [157, 206] for our task. Since the BLSTM-RNN has shown promising results in other handwriting analysis/recognition problems [14], we choose this classifier. Also, BLSTM-RNN is time efficient in comparison with DTW (Dynamic Time Warping) for shape matching [183].

In our approach, we employ the above 14 types of features. To feed into the BLSTM-RNN, in general, the features are categorized into 3 types: (a) vertical feature set f_V: $\{f_1, f_2, f_3, f_7, f_8, f_9, f_{11}\}$, (b) horizontal feature set f_H: $\{f_4, f_5, f_6, f_{10}, f_{12}\}$ and (c) special

feature set f_S: $\{f_{13}, f_{14}\}$. With these 3 sets of features (f_V, f_H and f_S), we employ 3 different BLSTM-RNNs (B_1, B_2 and B_3, respectively). The input layer of a BLSTM-RNN has N_d nodes, where 'N_d' is the number of features. Therefore, B_1, B_2 and B_3 have 7, 5 and 2 input nodes, respectively. The input nodes are connected with 2 distinct recurrent hidden layers, one for forward and another for the backward sequence. The output layer is also connected with both hidden layers. We primarily need to distinguish NE from non-NE words. Then four add-on special classes (*Wh-word*, *Th-word*, *Small-width* and *All-caps*: described in *Section 9.1.4*) are also considered. Therefore, the total number of classes is $N_c = 2 + 4 = 6$. At the output layer, one extra node is added for a non-word class due to noisy components/artifacts. Thus, the output layer of each of B_1, B_2 and B_3 contains $N_c + 1 = 7$ nodes. The details of BLSTM-RNN working principles can be found in [14]. The outputs of B_1, B_2 and B_3 are joined with a *fusion* scheme. Combining such classifiers, called *bagging* strategy, is applied for decreasing the error rate. The setup of BLSTM-RNN as the classifier for our problem is described in *Section 9.2.2*.

9.1.7 Post-processing

After classification, we check the positional occurrence (as described in *Section 9.1.4.2*) of the marked NEs. Some non-NEs, including *first* positional words, may be misclassified as NEs. We check with the special classes (*Wh-word*, *Th-word*, *Small-width* and *All-caps*; refer to *Section 9.1.4*) to separate the non-NEs. In this way, we attempt to reduce the false acceptance of the *first* positional non-NE words. We focus on minimizing the *genuine NE rejection* rate, sacrificing higher accuracy. It is more likely that we would find a subset of words, and those are most certainly the NEs. Therefore, we mark some words as potential NEs in this subset with a *degree of confidence* [102].

As mentioned earlier in this chapter, the NE reveals the content of a document. We also note that NEs, in certain writing, having many occurrences are indicative of the writing topic and thus assist in content understanding. Therefore, we count the frequency of a named entity also. For example, in Figure 9.1, the frequency of the named entity "Labor" is 2, and the frequency of "Wilkie" is also 2. Furthermore, we use the method of [14] to produce machine readable outputs of detected NE images.

9.2 Experiments and Discussion

Before discussing the experimental results, we begin with the description of the dataset used for evaluation of our scheme.

9.2.1 Database Employed

For our experiments, we used three handwritten offline datasets: a) George Washington database (GWdb) [240], b) Queensland State Archives database (QSAdb)[1] , c) IAM database (IAMdb) [232].

GWdb and QSAdb were historical text databases. Of those, the GWdb contained *20* handwritten pages of G. Washington. We took *66* pages from the QSAdb containing handwritten text only. Finally, the IAMdb contained recent data with *1539* handwritten pages written by 657 writers.

For the ground-truth generation of the NEs on such pages, a semi-automatic approach with human-intervention was employed. In addition, the manuscripts having publicly available transcripts were fed into the *Stanford Named Entity Recognizer* [123] for generating the ground truth.

9.2.2 Results and Evaluation

For training purposes, we considered four different sessions with random selections of 50%, 60%, 70% and 80% of the dataset to make four different classifiers (C_{1Bi}, C_{2Bi}, C_{3Bi} and C_{4Bi}) for each B_i, $i = 1,2,3$. This was a *dual layer bagging* strategy, i.e. bagging inside each bag. The classifier outputs (for both of the *inside bags*: $C_{jBi:\ j=1,2,3,4}$ and *outside bags*: $B_{i:\ i=1,2,3}$) were combined using a fusion strategy *Sum rule* over the confidence values for each class of each classifier. The sum rule was used because it exhibited better performance over some other fusion strategies such as the *max*, *min*, *median*, *majority vote*, *product* rule [117], etc. To avoid overfitting, 5-fold cross-validation was also used.

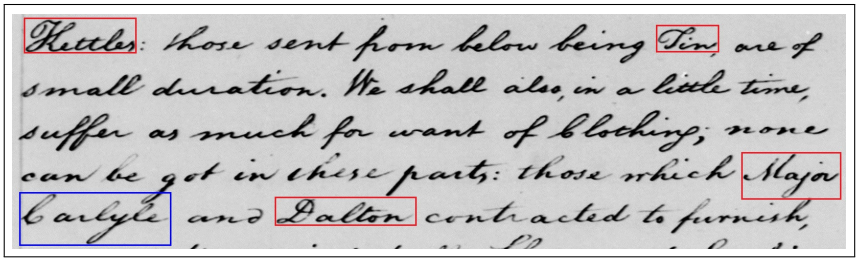
On overall dataset, *True Positive* (TP), *False Positive* (FP), and *False Negative* (FN) were obtained for named entity identification. Then, we calculated *Precision* (P), *Recall* (R), and *F-Measure* (FM) [2].

The overall P, R and FM are shown in Table 9.1.

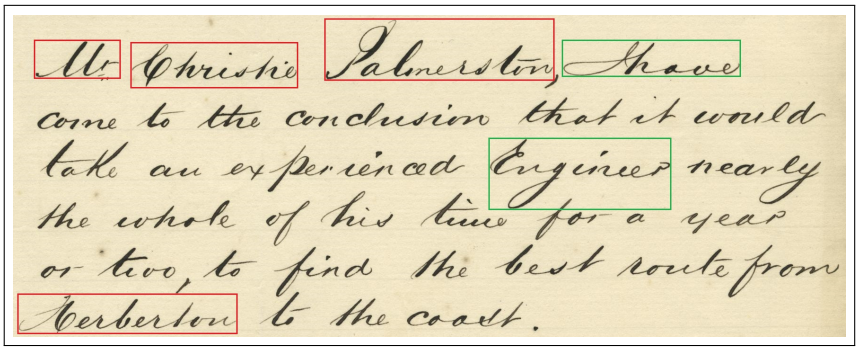
[1] "Queensland State Archives", Australia-4113. Online available at: `http://www.archivessearch.qld.gov.au/Search/BasicSearch.aspx`, retrieved on 31 Dec., 2018.

[2] $P = \frac{TP}{TP+FP}$, $R = \frac{TP}{TP+FN}$, $FM = \frac{2\times P\times R}{P+R}$

(a)



(b)



(c)

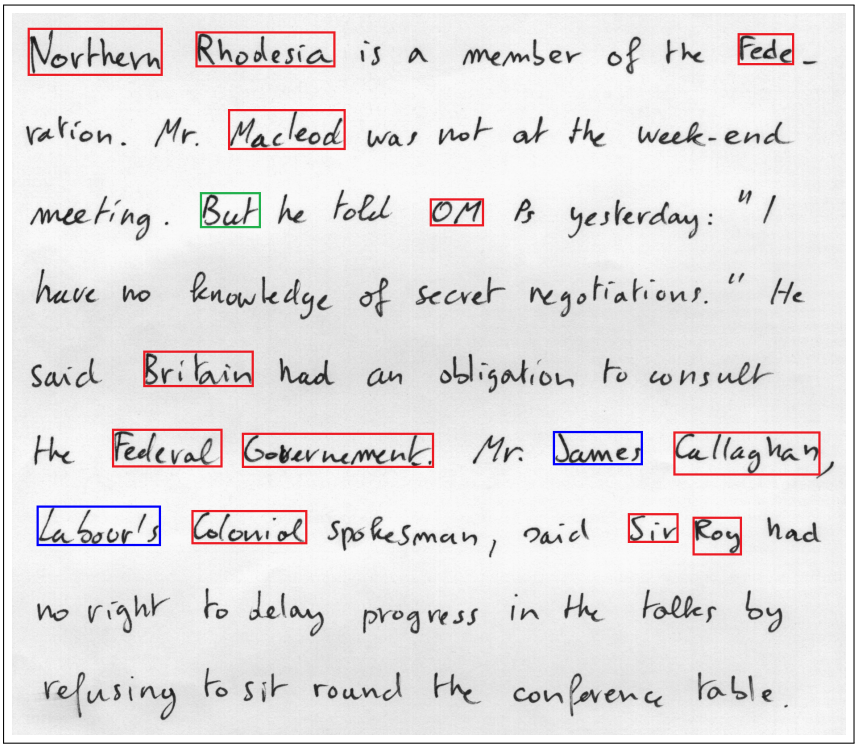


Figure 9.3: Qualitative result: NE identification in document images of (a) GWdb, (b) QSAdb, (c) IAMdb. Color boxes denote *true positive* (red), *false negative* (blue) and *false positive* (green) cases.

Table 9.1: Evaluation of named entity identification

Database	***Precision (%)***	***Recall (%)***	***F-Measure (%)***
GWdb	64.12	89.26	74.62
QSAdb	59.33	86.72	70.45
IAMdb	68.42	92.66	78.71

Since our aim is to reduce *genuine NE rejection*, i.e. minimizing FN, the *Recall* (R) should be high. The experimental results in Table 9.1 show fairly high *recall* values, predicting high *sensitivity*.

Table 9.2 shows the effectiveness of the employed features (refer to *Section 9.1.5*) on experimental results with respect to the F-Measure.

Table 9.2: Impact of employed features

Features	***F-Measure (%)***		
	GWdb	QSAdb	IAMdb
f_V: $\{f_1, f_2, f_3, f_7, f_8, f_9, f_{11}\}$	61.58	57.39	64.27
f_H: $\{f_4, f_5, f_6, f_{10}, f_{12}\}$	61.18	59.78	65.16
f_S: $\{f_{13}, f_{14}\}$	64.60	62.37	67.32

From Table 9.2, it may also be observed that *special weighted features* perform fairly well.

In Figure 9.3, we present some qualitative results of our system. Our method may identify a non-NE as an NE when a person writes a non-NE word starting with a capital letter (e.g., in Figure 9.3:b, the word "*Engineer*", marked in a green box). Conjoined cursive writing may lead to false acceptance (e.g., in Figure 9.3:b, the green box containing "*I have*"). Also, some *first* positioned words of a sentence can create false acceptance (e.g., in Figure 9.3:c, the word "*But*" marked by a green box).

Conversely, an NE starting with a capital letter, having a reasonably small width in the *upper zone*, can be rejected falsely (e.g., Figure 9.3:a, blue box containing an NE).

For comparison purposes, we did not find any other work on NE/CW recognition, related to ours, which was performed without employing a character/word recognition engine.

9.3 Summary

A named entity/content-revealing word recognizer for offline handwritten unstructured documents, without employing an OHCR and a linguistic model, is presented in this chapter. Experiments conducted on two historical datasets and one modern handwriting dataset have resulted in an average F-Measure of 74.59%. The proposed method is expected to work on some other *Latin* scripts, where an NE usually starts with a capital letter. Our method will not work on *Abugida* (e.g., Devanagari, Bangla, etc.) or, *Logographic* (e.g., Chinese, Japanese, etc.) script, where the concept of capital letters does not exist. Our future work will endeavor to make our system more accurate for English scripts. The present scheme can produce a set of NE/CW images and its machine-readable outputs to understand the content by human intervention. However, fully automated semantic understanding is yet to achieve, which will also be the scope of our future work.

CHAPTER 10

CONCLUSIONS AND FUTURE SCOPE

This chapter concludes the thesis and suggests some areas for future exploration. This thesis extends the area of handwriting research and works beyond the scope of optical (handwritten) character recognition engine. In fact, this research is a pioneering attempt of its kind to comprehend some new perceptions from handwriting.

In this thesis, we have studied the semantic understanding of handwritten images through three traits: *writer*, *writing stroke*, and *content*. Several artificial intelligence-based techniques have been adapted/proposed to automate this understanding and to build a human-like machine perception. Through a review of the thesis at the chapter level, it can be noted that writer understanding is investigated in *Chapter*s 2-5, whereas writing stroke understanding is studied in *Chapter*s 6-8, and the content understanding is examined in *Chapter* 9. We briefly summarize these chapters as follows.

– *Chapter 2* : In this chapter, we have proposed a new method to tackle writer verification. This method, at first, generates textural and allographic feature maps from a handwritten image, and then feeds the feature map into a convolutional neural network. Our scheme experimentally outperformed some state-of-the-art writer verification techniques.

– *Chapter 3* : Here, we have performed writer identification across scripts, i.e., training on one script while testing on another. We have taken the scenario of a multilingual country India, and investigated on cross-script writer identification using English and Bengali scripts. Here, we have employed some hand-crafted fea-

tures with various classifiers such as kNN, MLP, and SVM, and obtained promising results.

– *Chapter 4* : In this chapter, the writer identification/verification on intra-variable writing is studied. Some handcrafted and auto-derived features are used here. The handcrafted features are fed into SVM and the auto-derived features are fed into some state-of-the-art deep neural networks. We have performed a rigorous experiment for this study.

– *Chapter 5* : We have identified writer on handwritten pages those contain some struck-out text. Here, we have experimented the effect of struck-out texts on writer identification. For struck-out text detection, a CNN-SVM hybrid model is used. For writer identification, we use two separate models, (a) handcrafted features with SVM and (b) auto-derived CNN features with RNN. Experimentally, we have observed that struck-out text removal improves the writer identification performance.

– *Chapter 6* : Here, we work on legibility and aesthetic analysis of handwriting, to understand the handwriting clarity and beautiful appearances of writing strokes. We use auto-derived CNN features with RNN for legibility analysis, and handcrafted features with SVM for aesthetic analysis. We obtained some interesting results from the experiment.

– *Chapter 7* : We have studied the difficulties to write/read some handwritten characters. Here, a smaller version of the Inception Network has worked better than a handcrafted feature-based SVM model. We have also performed a correlation testing for reading and writing difficulty analysis.

– *Chapter 8* : In this chapter, we approach to identify idiosyncrasy in handwriting. We employ a smaller adaptation of Inception Network for idiosyncratic text identification. We also attempt to find the effect of idiosyncrasy in writer identification, where we use the SqueezeNet. We obtained some remarkable results.

– *Chapter 9* : We have attempted to understand the content of a handwritten specimen by analyzing some content-revealing words/named entities. For such word extraction, we propose a handcrafted feature-based BLSTM-RNN model and perform experiments on a contemporary and two historical writing databases.

Our research can provide an aid to the handwriting and forensic experts, biometric researchers, palaeographers, etc., for understanding the writer and writing strokes. Besides, this study has a positive social impact owing to judge the reading/writing difficulty. Therefore, some diseases (e.g., Parkinson's, Alzheimer's, Dysgraphia, Dyslexia, Tourette syndrome, etc.) which affect handwriting may be diagnosed by our work. Furthermore, this research analyzes legible and aesthetic writing, which can be helpful to a broad community and (pre-)schools to teach handwriting, as well as provides support to the educators/educational researchers. Additionally, our work provides an assistance to the librarians/curators/archival scientists for e-archiving the cultural heritage manuscripts by understanding the writer, writing strokes and writing content. Moreover, this research may be impactful for computer vision researchers considering the handwriting image as a visual pattern, and for the data scientists considering handwriting as data.

This research has plenty of future scopes, since it has led to a new handwriting-related research direction, i.e., the understanding of handwriting. In this thesis, we have studied some aspects of handwriting understanding. However, other aspects (refer to *Section* 1.1) can also be explored in the future. Some possible future research works are briefly mentioned as follows.

- Future scope of the writer understanding :
  - Our writer verification model of *Chapter* 2 is the hybridization of handcrafted feature map and CNN features. Instead of being a shallow neural network, this model has outperformed some state-of-the-art techniques. The productivity of using the shallow network can be analyzed in the future.
  - The writer identification across scripts is performed in *Chapter* 3, where alphabetic English script and alphasyllabary Bengali script are considered for the experiment. A cross-script analysis among several other scripts can be investigated in the future.
  - The writer identification/verification over intra-variable writing is studied in *Chapter* 4. Here, the intra-variability has mainly occurred due to speedy and exam writing. Some other intra-variability causing mechanical, physical, psychological factors can be taken into account for further experiment.
  - A writer is identified from a document that contains struck-out text, and presented in *Chapter* 5. On writer identification, the effect of the presence of struck-out text is also analyzed here. However, the writer inspection from struck-out style can be explored in the future.

- Future scope of the writing stroke understanding :
  - Our legibility/aesthetic analysis model of *Chapter* 6 attempts to understand the legible and aesthetic writing. Some effort can be made in the future to improve the accuracy of this model.
  - The model of *Chapter* 7 approaches to understanding the writing/reading difficulty of some Bengali conjunct characters. The experimentation can be extended in future for several other scripts.
  - The idiosyncrasy analysis module of *Chapter* 8 studies the individual peculiarity in handwriting. The impact of idiosyncratic writing on writer identification is also examined here. Some other perceptual impact can be thought of and scrutinized further.
- Future scope of the content understanding :
  - Our scheme of *Chapter* 9 can identify content-revealing words directly from a handwritten document image. However, the semantic relationship among these words is yet to be established and can be performed in the future.

Furthermore, development of a large dataset containing multiple handwritten samples from a wide-ranging and representative cohort of the population is an imminent need for performing research on handwriting understanding.

Finally, we believe that our work will encourage activities of document image processing in general, and handwriting analysis in particular.

"Handwriting is simply a pattern to a computer, but, to me, is a perceptual pattern."

— Chandranath Adak

BIBLIOGRAPHY

[1] A. A. DMOUR, R. A. ZITAR, *Arabic Writer Identification based on Hybrid Spectral-Statistical Measures*, Journal of Experimental & Theoretical Artificial Intelligence, 19 (2007), pp. 307–332.

[2] A. ALAEI, U. PAL, P. NAGABHUSHAN, *Dataset and Ground Truth for Handwritten Text in Four Different Scripts*, Int. Journal of Pattern Recognition and Artificial Intelligence (IJPRAI), 26 (2012), p. #1253001.

[3] A. BENSEFIA, A. NOSARY, T. PAQUET, L. HEUTTE, *Writer Identification by Writer's Invariants*, Int. Workshop on Frontiers in Handwriting Recognition (IWFHR), (2002), pp. 274–279.

[4] A. BENSEFIA, T. PAQUET, L. HEUTTE, *Handwriting Analysis for Writer Verification*, IWFHR, (2004), pp. 196–201.

[5] ——, *A Writer Identification and Verification System*, Pattern Recognition Letters, 26 (2005), pp. 2080–2092.

[6] A. BHARDWAJ, A. SINGH, H. SRINIVASAN, S. N. SRIHARI, *On the Use of Lexeme Features for Writer Verification*, Int. Conf. on Document Analysis and Recognition (ICDAR), (2007), pp. 1088–1092.

[7] A. BRINK, H. VAN DER KLAUW, L. SCHOMAKER, *Automatic Removal of Crossed-Out Handwritten Text and the Effect on Writer Verification and Identification*, Document Recognition and Retrieval (DRR) XV, #68150A, (2008).

[8] A. BRINK, L. SCHOMAKER, M. BULACU, *Towards Explainable Writer Verification and Identification using Vantage Writers*, ICDAR, (2007), pp. 824–828.

[9] A. BRINK, R. M. J. NIELS, R. A. VAN BATENBURG, C. E. VAN DEN HEUVEL, L. R. B. SCHOMAKER, *Towards Robust Writer Verification by Correcting Unnatural Slant*, Pattern Recognition Letters, 32 (2011), pp. 449–457.

[10] A. FAHAD ET AL., *A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis*, IEEE Trans. on Emerging Topics in Computing, 2 (2014), pp. 267–279.

[11] A. FORNES, A. DUTTA, A. GORDO, J. LLADOS, *The ICDAR 2011 Music Scores Competition: Staff Removal and Writer Identification*, ICDAR, (2011), pp. 1511–1515.

[12] ——, *CVC-MUSCIMA: A Ground Truth of Handwritten Music Score Images for Writer Identification and Staff Removal*, Int. Journal on Document Analysis and Recognition (IJDAR), 15 (2012), pp. 243–251.

[13] A. GORDO, A. FORNES, E. VALVENY, *Writer Identification in Handwritten Musical Scores with Bags of Notes*, Pattern Recognition, 46 (2013), pp. 1337–1345.

[14] A. GRAVES, M. LIWICKI, S. FERNANDEZ, R. BERTOLAMI, H. BUNKE, J. SCHMIDHUBER, *A Novel Connectionist System for Unconstrained Handwriting Recognition*, IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), 31 (2009), pp. 855–868.

[15] A. HASSAINE, S. A. MAADEED, *ICFHR 2012 Competition on Writer Identification Challenge 2: Arabic Scripts*, Int. Conf. on Frontiers in Handwriting Recognition (ICFHR), (2012), pp. 835–840.

[16] A. HASSAINE, S. A. MAADEED, J. M. ALJAAM, A. JAOUA, A. BOURIDANE, *The ICDAR 2011 Arabic Writer Identification Contest*, ICDAR, (2011), pp. 1470–1474.

[17] A. IMDAD, S. BRES, V. EGLIN, C. R. MORENO, H. EMPTOZ, *Writer Identification using Steered Hermite Features and SVM*, ICDAR, (2007), pp. 839–843.

[18] A. J. NEWELL, L. D. GRIFFIN, *Writer Identification using Oriented Basic Image Features and the Delta Encoding*, Pattern Recognition, 47 (2014), pp. 2255–2265.

[19] A. K. JAIN, A. ROSS, S. PRABHAKAR, *An Introduction to Biometric Recognition*, IEEE Trans. on Circuits and Systems for Video Technology, 14 (2004), pp. 4–20.

[20] A. KRIZHEVSKY, I. SUTSKEVER, G. E. HINTON, *ImageNet Classification with Deep Convolutional Neural Networks*, Int. Conf. on Neural Information Processing Systems (NIPS), 1 (2012), pp. 1097–1105.

[21] A. L. WONG, A. M. HAITH, J. W. KRAKAUER, *Motor Planning*, Neuroscientist, 21 (2015), pp. 385–398.

[22] A. M. WING, A. D. BADDELEY, *A Simple Measure of Handwriting as an Index of Stress*, Bulletin of the Psychonomic Society, 11 (1978), pp. 245–246.

[23] A. MAJUMDAR, P. KRISHNAN, C. V. JAWAHAR, *Visual Aesthetic Analysis for Handwritten Document Images*, ICFHR, (2016), pp. 423–428.

[24] A. MEZGHANI, S. KANOUN, M. KHEMAKHEM, H. E. ABED, *A Database for Arabic Handwritten Text Image Recognition and Writer Identification*, ICFHR, (2012), pp. 399–402.

[25] A. NAFTALI, *Behavior Factors in Handwriting Identification*, The Journal of Criminal Law, Criminology and Police Science, 56 (1965), pp. 528–539.

[26] A. NICOLAOU, A. D. BAGDANOV, M. LIWICKI, D. KARATZAS, *Sparse Radial Sampling LBP for Writer Identification*, ICDAR, (2015), pp. 716–720.

[27] A. NICOLAOU, M. LIWICKI, R. INGOLD, *Oriented Local Binary Patterns for Writer Identification*, ICDAR Int. Workshop on Automated Forensic Handwriting Analysis, (2013), pp. 15–20.

[28] A. SCHLAPBACH, H. BUNKE, *Using HMM based Recognizers for Writer Identification and Verification*, IWFHR, (2004), pp. 167–172.

[29] ——, *Writer Identification using an HMM-based Handwriting Recognition System: To Normalize the Input or Not?*, International Graphonomics Society Conference (IGS), (2005), pp. 138–142.

[30] ——, *Off-line Writer Identification using Gaussian Mixture Models*, Int. Conference on Pattern Recognition (ICPR), (2006), pp. 992–995.

[31] ——, *Off-Line Writer Verification: A Comparison of a Hidden Markov Model (HMM) and a Gaussian Mixture Model (GMM) based System*, ICFHR, (2006), p. #1025.

[32] ——, *A Writer Identification and Verification System using HMM Based Recognizers*, Pattern Analysis and Applications, 10 (2007), pp. 33–43.

[33] A. SCHLAPBACH, V. KILCHHERR, H. BUNKE, *Improving Writer Identification by Means of Feature Selection and Extraction*, ICDAR, (2005), pp. 131–135.

[34] A. W. ELLIS, *Reading, Writing and Dyslexia: A Cognitive Analysis*, Psychology Press, New York, (1993).

[35] B. B. CHAUDHURI, *Learning an Indian Abugida Script : Bangla*, International Graphonomics Society Conference (IGS), (2011), pp. 22–25.

[36] B. B. CHAUDHURI, C. ADAK, *An Approach for Detecting and Cleaning of Struck-out Handwritten Text*, Pattern Recognition, 61 (Jan. 2017), pp. 282–294.

[37] B. B. CHAUDHURI, U. PAL, *A Complete Printed Bangla OCR System*, Pattern Recognition, 31 (1998), pp. 531–549.

[38] B. J. NAVYA, P. SHIVAKUMARA, G. C. SWETHA, S. ROY, D. S. GURU, U. PAL, T. LU, *Adaptive Multi-Gradient Kernels for Handwritting based Gender Identification*, ICFHR, (2018), pp. 392–397.

[39] B. MCCUNE, J. B. GRACE, *Analysis of Ecological Communities*, MjM Software, Gleneden Beach, Oregon, USA, 2002, ch. 6.

[40] B. V. B. DONSKY, *Trends in Elementary Writing Instruction, 1900-1959*, Language Arts, 61 (1984), pp. 795–803.

[41] B. ZHANG, S. N. SRIHARI, S. LEE, *Individuality of Handwritten Characters*, ICDAR, (2003), pp. 1086–1090.

[42] C. ADAK, B. B. CHAUDHURI, *Extraction of Doodles and Drawings from Manuscripts*, Int. Conf. on Pattern Recognition and Machine Intelligence (PReMI), LNCS #8251, (2013), pp. 515–520.

[43] ——, *An Approach of Strike-through Text Identification from Handwritten Documents*, ICFHR, (2014), pp. 643–648.

[44] ——, *Writer Identification from Offline Isolated Bangla Characters and Numerals*, ICDAR, (2015), pp. 486–490.

[45] C. ADAK, B. B. CHAUDHURI, M. BLUMENSTEIN, *Offline Cursive Bengali Word Recognition using CNNs with a Recurrent Model*, ICFHR, (2016), pp. 429–434.

[46] ——, *Writer Identification by Training on One Script but Testing on Another*, ICPR, (2016), pp. 1148–1153.

[47] ——, *A Study on Writer Identification and Verification from Intra-variable Individual Handwriting*, arXiv.org: CoRR abs/1708.03361, (2017).

[48] ——, *Impact of Struck-out Text on Writer Identification*, International Joint Conference on Neural Networks (IJCNN), (2017), pp. 1465–1471.

[49] ——, *Legibility and Aesthetic Analysis of Handwriting*, ICDAR, (2017), pp. 175–182.

[50] C. B. OLSON, R. LAND, *A Cognitive Strategies Approach to Reading and Writing Instruction for English Language Learners in Secondary School*, Research in the Teaching of English, 41 (2007), pp. 269–303.

[51] C. D. MANNING, P. RAGHAVAN, H. SCHüTZE, *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[52] C. DJEDDI, I. SIDDIQI, L. S. MESLATI, A. ENNAJI, *Codebook for Writer Characterization: A Vocabulary of Patterns or a Mere Representation Space?*, ICDAR, (2013), pp. 423–427.

[53] C. DJEDDI, S. A. MAADEED, A. GATTAL, I. SIDDIQI, A. ENNAJI, H. E. ABED, *ICFHR2016 Competition on Multi-script Writer Demographics Classification using "QUWI" Database*, ICFHR, (2016), pp. 602–606.

[54] C. DJEDDI, S. A. MAADEED, A. GATTAL, I. SIDDIQI, L. S. MESLATI, H. E. ABED, *ICDAR2015 competition on Multi-script Writer Identification and Gender Classification using "QUWI" Database*, ICDAR, (2015), pp. 1191–1195.

[55] C. HERTEL, H. BUNKE, *A Set of Novel Features for Writer Identification*, Int. Conf. on Audio and Video-based Biometric Person Authentication, (2003), pp. 679–687.

[56] C. I. TOMAI, B. ZHANG, S. N. SRIHARI, *Discriminatory Power of Handwritten Words for Writer Recognition*, ICPR, 2 (2004), pp. 638–641.

[57] C. L. LIU, C. Y. SUEN, *A New Benchmark on the Recognition of Handwritten Bangla and Farsi Numeral Characters*, Pattern Recognition, 42 (2009), pp. 3287–3295.

[58] C. L. LIU, F. YIN, D. WANG, Q. WANG, *CASIA Online and Offline Chinese Handwriting Databases*, ICDAR, (2011), pp. 37–41.

[59] C. L. LIU, R. W. DAI, Y. J. LIU, *Extracting Individual Features from Moments for Chinese Writer Identification*, ICDAR, I (1995), pp. 438–441.

[60] C. O. A. FREITAS, L. S. OLIVEIRA, R. SABOURIN, F. BORTOLOZZI, *Brazilian Forensic Letter Database*, IWFHR, (2008), p. #1011.

[61] C. RAMAIAH, V. GOVINDARAJU, *A Hierarchical Framework for Accent based Writer Identification*, Int. Workshop on Document Analysis Systems (DAS), (2014), pp. 21–25.

[62] C. SZEGEDY, V. VANHOUCKE, S. IOFFE, J. SHLENS, Z. WOJNA, *Rethinking the Inception Architecture for Computer Vision*, arXiv:1512.00567, (2015).

[63] C. SZEGEDY, W. LIU, Y. JIA, P. SERMANET, S. REED, D. ANGUELOV, D. ERHAN, V. VANHOUCKE, A. RABINOVICH, *Going Deeper with Convolutions*, arXiv:1409.4842, (2014).

[64] D. BERTOLINI, L. S. OLIVEIRA, E. JUSTINO, R. SABOURIN, *Texture-based Descriptors for Writer Identification and Verification*, Expert Systems with Applications, 40 (2013), pp. 2069–2080.

[65] D. F. MOTA, P. RIBA, A. FORNES, J. LLADOS, *On the Influence of Key Point Encoding for Handwritten Word Spotting*, ICFHR, (2014), pp. 476–481.

[66] D. FECKER, A. ASI, W. PANTKE, V. MARGNER, J. E. SANA, T. FINGSCHEIDT, *Document Writer Analysis with Rejection for Historical Arabic Manuscripts*, ICFHR, (2014), pp. 743–748.

[67] D. NADEAU, S. SEKINE, *A Survey of Named Entity Recognition and Classification*, J. Lingvisticae Investigationes, John Benjamins Pub. Co., 30 (2007), pp. 3–26.

[68] D. PARASKEVAS, S. GRITZALIS, E. KAVALLIERATOU, *Writer Identification using a Statistical and Model Based Approach*, ICFHR, (2014), pp. 589–594.

[69] D. STARCH, *The Measurements of Handwriting*, The Journal of Educational Psychology, 4 (1913), pp. 445–464.

[70] D. TUGANBAEV, D. DERIAGUINE, *Method of Stricken-out Character Recognition in Handwritten Text*, Patent: US 8472719 B2, (2013).

[71] E. AHMED, M. JONES, T. K. MARKS, *An Improved Deep Learning Architecture for Person Re-identification*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), (2015), pp. 3908–3916.

[72] E. GROSICKI, M. CARRE, J. M. BRODIN, E. GEOFFROIS, *Results of the RIMES Evaluation Campaign for Handwritten Mail Processing*, ICDAR, (2009), pp. 941–945.

[73] E. HOFFER, N. AILON, *Deep Metric Learning using Triplet Network*, arXiv: 1412.6622, (2014).

[74] E. KAVALLIERATOU, N. FAKOTAKIS, G. K. KOKKINAKIS, *Slant Estimation Algorithm for OCR Systems*, Pattern Recognition, 34 (2001), pp. 2515–2522.

[75] E. N. ZOIS, V. ANASTASSOPOULOS, *Morphological Waveform Coding for Writer Identification*, Pattern Recognition, 33 (2000), pp. 385–398.

[76] E. R. PLATTOR, E. S. WOESTEHOFF, *The Relationship between Reading Manuscript and Cursive Writing*, Elementary English, 44 (1967), pp. 50–52.

[77] E. WIGNER, *On the Quantum Correction for Thermodynamic Equilibrium*, Physical Review, 40 (1932), pp. 749–759.

[78] F. A. KHAN, M. A. TAHIR, F. KHELIFI, A. BOURIDANE, R. ALMOTAERYI, *Robust Off-line Text Independent Writer Identification using Bagged Discrete Cosine Transform Features*, Expert Systems with Applications, 71 (2017), pp. 404–415.

[79] F. ASICIOGLU, N. TURAN, *Handwriting Changes Under the Effect of Alcohol*, Forensic Science International, 132 (2003), pp. 201–210.

[80] F. CHOLLET, *Xception: Deep Learning with Depthwise Separable Convolutions*, arXiv:1610.02357v3, (2016).

[81] F. KLEBER, S. FIEL, M. DIEM, R. SABLATNIG, *CVL-Database: An Off-line Database for Writer Retrieval, Writer Identification and Word Spotting*, ICDAR, (2013), pp. 560–564.

[82] F. N. FREEMAN, *An Analytical Scale for Judging Handwriting*, The Elementary School Journal, 15 (1915), pp. 432–441.

[83] F. N. IANDOLA, S. HAN, M. W. MOSKEWICZ, K. ASHRAF, W. J. DALLY, K. KEUTZER, *SqueezeNet: AlexNet-level Accuracy with 50x Fewer Parameters and <0.5 MB Model Size*, arXiv:1602.07360, (2016).

[84] F. NEJAD, M. RAHMATI, *A New Method for Writer Identification and Verification Based on Farsi/Arabic Handwritten Texts*, ICDAR, (2007), pp. 829–833.

[85] F. SHAHABI, M. RAHMATI, *A New Method for Writer Identification of Handwritten Farsi Documents*, ICDAR, (2009), pp. 426–430.

[86] F. SLIMANE, V. MARGNER, *A New Text-Independent GMM Writer Identification System Applied to Arabic Handwriting*, ICFHR, (2014), pp. 708–713.

[87] F. ZHAO, H. Z. LU, Z. Y. ZHANG, *Real-time Single-pass Connected Components Analysis Algorithm*, EURASIP Journal on Image and Video Processing, 21 (2013), pp. 1–10.

[88] G. D. ZHOU, J. SU, *Named Entity Recognition using an HMM-based Chunk Tagger*, Annual Meeting on Association for Computational Linguistics (ACL), (2002), pp. 473–480.

[89] G. E. HINTON, N. SRIVASTAVA, A. KRIZHEVSKY, I. SUTSKEVER, R. R. SALAKHUTDINOV, *Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors*, arXiv: 1207.0580, (2012).

[90] G. KOCH, *Siamese Neural Networks for One-Shot Image Recognition*, M.S. Thesis, University of Toronto, (2015).

[91] G. LOULOUDIS, B. GATOS, N. STAMATOPOULOS, *ICFHR 2012 Competition on Writer Identification Challenge 1: Latin/Greek Documents*, ICFHR, (2012), pp. 829–834.

[92] G. LOULOUDIS, B. GATOS, N. STAMATOPOULOS, A. PAPANDREOU, *ICDAR 2013 Competition on Writer Identification*, ICDAR, (2013), pp. 1397–1401.

[93] G. LOULOUDIS, N. STAMATOPOULOS, B. GATOS, *ICDAR 2011 Writer Identification Contest*, ICDAR, (2011), pp. 1475–1479.

[94] G. R. BALL, S. N. SRIHARI, R. STRITMATTER, *Writer Verification of Historical Documents among Cohort Writers*, ICFHR, (2010), pp. 314–319.

[95] G. SALTON, *The SMART Retrieval System - Experiments in Automatic Document Processing*, Prentice-Hall Inc., (1971).

[96] G. ZHU, T. J. BETHEA, V. KRISHNA, *Extracting Relevant Named Entities for Automated Expense Reimbursement*, ACM Conf. on Knowledge, Discovery and Data Mining (KDD), (2007), pp. 1004–1012.

[97] H. A. GLUCKSMAN, *Classification of Mixed-font Alphabets by Characteristic Loci*, Annual IEEE Computer Conference, (1967), pp. 138–141.

[98] H. E. S. SAID, T. N. TAN, K. D. BAKER, *Personal Identification based on Handwriting*, Pattern Recognition, 33 (2000), pp. 149–160.

[99] H. HILTON, *Plane Algebraic Curves*, Oxford, 1920, ch. II, pp. 18–36.

[100] H. KAMEYA, S. MORI, R. OKA, *Figure-Based Writer Verification by Matching between an Arbitrary Part of Registered Sequence and an Input Sequence Extracted from On-Line Handwritten Figures*, ICDAR, (2003), pp. 985–989.

[101] H. SRINIASAN, S. KABRA, C. HUANG, S. N. SRIHARI, *On Computing Strength of Evidence for Writer Verification*, ICDAR, (2007), pp. 844–848.

[102] H. ZARAGOZA, F. D'ALCHE-BUC, *Confidence Measures for Neural Network Classifiers*, Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, 1 (1998), pp. 886–893.

[103] I. DINSTEIN, Y. SHAPIRA, *Ancient Hebraic Handwriting Identification with Run-Length Histograms*, IEEE Trans. on Systems, Man, and Cybernetics, 12 (1982), pp. 405–409.

[104] I. J. KIM, J. H. KIM, *Statistical Character Structure Modeling and Its Application to Handwritten Chinese Character Recognition*, IEEE Trans. on PAMI, 25 (2003), pp. 1422–1436.

[105] I. SIDDIQI, C. DJEDDI, A. RAZA, L. S. MESLATI, *Automatic Analysis of Handwriting for Gender Classification*, Pattern Analysis and Applications, 18 (2014), pp. 887–899.

[106] I. SIDDIQI, N. VINCENT, *Writer Identification in Handwritten Documents*, ICDAR, (2007), pp. 108–112.

[107] ——, *A Set of Chain Code Based Features for Writer Recognition*, ICDAR, (2009), pp. 981–985.

[108] I. YOSHIMURA, M. YOSHIMURA, *Off-line Writer Verification using Ordinary Characters as the Object*, Pattern Recognition, 24 (1991), pp. 909–915.

[109] J. ARLANDIS, J.C.P. CORTES, J. CANO, *Rejection Strategies and Confidence Measures for a k-NN Classifier in an OCR Task*, ICPR, 1 (2002), pp. 576–579.

[110] J. BEHRENDT, *Alzheimer's Disease and Its Effect on Handwriting*, Journal of Forensic Sciences, 29 (1984), pp. 87–91.

[111] J. BROMLEY, I. GUYON, Y. LECUN, E. SäCKINGER, R. SHAH, *Signature Verification using a "Siamese" Time Delay Neural Network*, NIPS, (1993), pp. 737–744.

[112] J. CHEN, D. P. LOPRESTI, *Exploiting Ruling Line Artifacts in Writer Identification*, ICPR, (2012), pp. 3737–3740.

[113] ——, *Alternatives for Page Skew Compensation in Writer Identification*, ICDAR, (2013), pp. 927–931.

[114] J. CHEN, D. P. LOPRESTI, G. NAGY, *Conservative Preprocessing of Document Images*, IJDAR, 19 (2016), pp. 321–333.

[115] J. CHEN, W. CHENG, D. P. LOPRESTI, *Using Perturbed Handwriting to Support Writer Identification in the Presence of Severe Data Constraints*, DRR XVIII, 7874 (2011).

[116] J. J. SCHNEIDER, *Contexts, Genres, and Imagination: An Examination of the Idiosyncratic Writing Performances of Three Elementary Children within Multiple Contexts of Writing Instruction*, Research in the Teaching of English, 37 (2003), pp. 329–379.

[117] J. KITTLER, M. HATEF, R. P. W. DUIN, J. MATAS, *On Combining Classifiers*, IEEE Trans. on PAMI, 20 (1998), pp. 226–239.

[118] J. L. MYERS, A. D. WELL, R. F. LORCH JR., *Research Design and Statistical Analysis*, Mahwah, N.J. : Lawrence Erlbaum Associates, (2003).

[119] J. MATHYER, *The Influence of Writing Instruments on Handwriting and Signatures*, The Journal of Criminal Law, Criminology and Police Science, 60 (1969), pp. 102–112.

[120] J. MICHELL, *Who Wrote Shakespeare?*, Thames and Hudson Ltd., London, 1996.

[121] J. MILGRAM, M. CHERIET, R. SABOURIN, *"One Against One" or "One Against All": Which One is Better for Handwriting Recognition with SVMs?*, IWFHR, (2006).

[122] J. NEYMAN, E. PEARSON, *On the Problem of the Most Efficient Tests of Statistical Hypotheses*, Philosophical Transactions of the Royal Society of London, series A, 231 (1933), pp. 289–337.

[123] J. R. FINKEL, T. GRENAGER, C. MANNING, *Incorporating Non-Local Information into Information Extraction Systems by Gibbs Sampling*, Annual Meeting on Association for Computational Linguistics (ACL), (2005), pp. 363–370.

[124] J. SCHMIDHUBER, *Deep Learning in Neural Networks: An Overview*, Neural Networks, 61 (2015), pp. 85–117.

[125] J. TAN, J. H. LAI, W. S. ZHENG, M. ZHONG, *Chinese Handwritten Writer Identification based on Structure Features and Extreme Learning Machine*, Int. Workshop on Automated Forensic Handwriting Analysis, (2013), pp. 21–25.

[126] J. TAN, N. BI, C. Y. SUEN, N. NOBILE, *Multi-Feature Selection of Handwriting for Gender Identification using Mutual Information*, ICFHR, (2016), pp. 578–583.

[127] J. WALTON, *Handwriting Changes Due to Aging and Parkinson's Syndrome*, Forensic Science International, 88 (1997), pp. 197–214.

[128] K. AMEND, M. S. RUIZ, *Handwriting Analysis: The Complete Basic Book*, The Career Press, NJ 07417, 1980.

[129] K. DUAN, S. S. KEERTHI, A. N. POO, *Evaluation of Simple Performance Measures for Tuning SVM Hyperparameters*, Neurocomputing, 51 (2003), pp. 41–59.

[130] K. DUTTA, P. KRISHNAN, M. MATHEW, C. V. JAWAHAR, *Offline Handwriting Recognition on Devanagari using a New Benchmark Dataset*, DAS, (2018), pp. 25–30.

[131] K. H. BRODERSEN, C. S. ONG, K. E. STEPHAN, J. M. BUHMANN, *The Balanced Accuracy and Its Posterior Distribution*, ICPR, (2010), pp. 3121–3124.

[132] K. HE, X. ZHANG, S. REN, J. SUN, *Deep Residual Learning for Image Recognition*, CVPR, (2016), pp. 770–778.

[133] K. M. KOPPENHAVER, *Factors That Cause Changes in Handwriting*, Humana Press, New Jersey, 2007, ch. 3 of "Forensic Document Examination: Principles and Practice", pp. 27–36.

[134] K. NTIROGIANNIS, B. GATOS, I. PRATIKAKIS, *A Combined Approach for the Binarization of Handwritten Document Images*, Pattern Recognition Letters, 35 (2014), pp. 3–15.

[135] K. R. BANDI, S. N. SRIHARI, *Writer Demographic Classification using Bagging and Boosting*, International Graphonomics Society Conference (IGS), (2005), pp. 133–137.

[136] K. SIMONYAN, A. ZISSERMAN, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, arXiv:1409.1556, (2014).

[137] K. TSELIOS, E.N. ZOIS, A. NASSIOPOULOS, S. KARABETSOS, G. ECONOMOU, *Automated Off-Line Writer Verification using Short Sentences and Grid Features*, Int. Workshop on Automated Forensic Handwriting Analysis, (2011), pp. 21–25.

[138] L. DU, X. YOU, H. XU, Z. GAO, Y. TANG, *Wavelet Domain Local Binary Pattern Features For Writer Identification*, ICPR, (2010), pp. 3691–3694.

[139] L. L. SULEM, A. VINCIARELLI, *HMM-based Offline Recognition of Handwritten Words Crossed Out with Different Kinds of Strokes*, ICFHR, (2008), pp. 70–75.

[140] L. LAM, S. W. LEE, C. Y. SUEN, *Thinning Methodologies - A Comprehensive Survey*, IEEE Trans. on PAMI, 14 (1992), pp. 869–885.

[141] L. MARCHESOTTI, N. MURRAY, F. PERRONNIN, *Discovering Beautiful Attributes for Aesthetic Image Analysis*, International Journal of Computer Vision, 113 (2015), pp. 246–266.

[142] L. NANNI, S. GHIDONI, S. BRAHNAM, *Handcrafted vs. Non-Handcrafted Features for Computer Vision Classification*, Pattern Recognition, 71 (2017), pp. 158–172.

[143] L. QUANT, *Factors Affecting the Legibility of Handwriting*, The Journal of Experimental Education, 14 (1946), pp. 297–316.

[144] L. SCHOMAKER, *Advances in Writer Identification and Verification*, ICDAR, 2 (2007), pp. 1268–1273.

[145] ——, *Writer Identification and Verification*, Springer, London, 2008, ch. 13 of "Advances in Biometrics: Sensors, Algorithms and Systems", pp. 247–264.

[146] L. SCHOMAKER, M. BULACU, *Automatic Writer Identification using Connected-Component Contours and Edge-Based Features of Uppercase Western Script*, IEEE Trans. on PAMI, 26 (2004), pp. 787–798.

[147] L. XING, Y. QIAO, *DeepWriter: A Multi-Stream Deep CNN for Text-Independent Writer Identification*, ICFHR, (2016), pp. 584–589.

[148] L. YI, Z. YEFENG, D. DOERMANN, *Detecting Text Lines in Handwritten Documents*, ICPR, (2006), pp. 1030–1033.

[149] M. BULACU, L. SCHOMAKER, *Text-Independent Writer Identification and Verification using Textural and Allographic Features*, IEEE Trans. on PAMI, 29 (2007), pp. 701–717.

[150] M. BULACU, L. SCHOMAKER, A. BRINK, *Text-Independent Writer Identification and Verification on Offline Arabic Handwriting*, ICDAR, (2007), pp. 769–773.

[151] M. G. LODICO, D. T. SPAULDING, K. H. VOEGTLE, *Methods in Educational Research: From Theory to Practice*, Jossey-Bass: A Wiley Imprint, USA, 2006.

[152] M. K. CARTER, *Manuscript Writing or Cursive Writing?*, The Journal of Education, 136 (1953), pp. 2–3.

[153] M. K. SHARMA, V. P. DHAKA, *Offline Scripting-Free Author Identification based on Speeded-Up Robust Features*, IJDAR, 18 (2015), pp. 303–316.

[154] M. LIN, Q. CHEN, S. YAN, *Network in Network*, CoRR, abs/1312.4400, (2013).

[155] M. N. ABDI, M. KHEMAKHEM, *A Model-based Approach to Offline text-independent Arabic Writer Identification and Verification*, Pattern Recognition, 48 (2015), pp. 1890–1903.

[156] M. PECHWITZ, S. MADDOURI, V. MARGNER, N. ELLOUZE, H. AMIRI, *IFN/ENIT-Database of Handwritten Arabic Words*, CIFED, (2002), pp. 129–136.

[157] M. SCHUSTER, K. K. PALIWAL, *Bidirectional Recurrent Neural Networks*, IEEE Transactions on Signal Processing, 45 (1997), pp. 2673–2681.

[158] M. TAOUK, M. COLTHEART, *The Cognitive Processes Involved in Learning to Read in Arabic*, Reading and Writing: An Interdisciplinary Journal, 17 (2004), pp. 27–57.

[159] M. WIROTIUS, A. SEROPIAN, N. VINCENT, *Writer Identification from Gray Level Distribution*, ICDAR, (2003), pp. 1168–1172.

[160] M. Z. ALOM ET AL., *The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches*, arXiv:1803.01164, (2018).

[161] N. BOUADJENEK, H. NEMMOUR, Y. CHIBANI, *Age, Gender and Handedness Prediction from Handwriting using Gradient Features*, ICDAR, (2015), pp. 1116–1120.

[162] N. DALAL, B. TRIGGS, *Histograms of Oriented Gradients for Human Detection*, CVPR, 1 (2005), pp. 886–893.

[163] N. OTSU, *A Threshold Selection Method from Gray-Level Histograms*, IEEE Trans. on Systems, Man and Cybernetics, 9 (1979), pp. 62–66.

[164] N. SRIVASTAVA, G.E. HINTON, A. KRIZHEVSKY, I. SUTSKEVER, R. SALAKHUTDINOV, *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*, Journal of Machine Learning Research (JMLR), 15 (2014), pp. 1929–1958.

[165] N. STAMATOPOULOS, B. GATOS, A. KESIDIS, *Automatic Borders Detection of Camera Document Images*, Int. Workshop on Camera-Based Document Analysis and Recognition, (2007), pp. 71–78.

[166] N. STAMATOPOULOS, B. GATOS, G. LOULOUDIS, U. PAL A. ALAEI, *ICDAR 2013 Handwriting Segmentation Contest*, ICDAR, (2013), pp. 1402–1406.

[167] O. HILTON, *Effects of Writing Instruments on Handwriting Details*, Journal of Forensic Sciences, 29 (1984), pp. 80–86.

[168] O. KIRLI, M. B. GULMEZOGLU, *Automatic Writer Identification from Text Line Images*, IJDAR, 15 (2012), pp. 85–99.

[169] O. RUSSAKOVSKY ET AL., *ImageNet Large Scale Visual Recognition Challenge*, International Journal of Computer Vision, 115 (2015), pp. 211–252.

[170] P. EMERSON, *The Original Borda Count and Partial Voting*, Social Choice and Welfare, 40 (2013), pp. 353–358.

[171] P. J. SUTANTO, G. LEEDHAM, V. PERVOUCHINE, *Study of the Consistency of Some Discriminatory Features Used by Document Examiners in the Analysis of Handwritten Letter 'a'*, ICDAR, (2003), pp. 1091–1095.

[172] P. P. ROY, A. K. BHUNIA, A. DAS, P. DEY, U. PAL, *HMM-based Indic Handwritten Word Recognition using Zone Segmentation*, Pattern Recognition, 60 (2016).

[173] P. T. DANIELS, W. BRIGHT, EDS., *The World's Writing Systems*, Oxford University Press, New York, 1996.

[174] P. TRIPATHY, B. CHANDA, B. B. CHAUDHURI, *Writer Recognition by Analyzing Word Level Features of Handwritten Documents*, ICAPR, World Scientific, (2007), pp. 73–78.

[175] P. YE, D. DOERMANN, *Document Image Quality Assessment: A Brief Survey*, ICDAR, (2013), pp. 723–727.

[176] Q. H. THU, M. N. GARCIA, F. SPERANZA, P. CORRIVEAU, A. RAAKE, *Study of Rating Scales for Subjective Quality Assessment of High-Definition Video*, IEEE Trans. on Broadcasting, 57 (2011), pp. 1–14.

[177] R. A. HUBER, A. M. HEADRICK, *Handwriting Identification: Facts and Fundamentals*, CRC Press, 1999.

[178] R. D. BANERJI, *The Origin of the Bengali Script*, University of Calcutta, 1919.

[179] R. DATTA, D. JOSHI, J. LI, J. Z. WANG, *Studying Aesthetics In Photographic Images using a Computational Approach*, European Conference on Computer Vision, (2006), pp. 288–301.

[180] R. HADSELL, S. CHOPRA, Y. LECUN, *Dimensionality Reduction by Learning an Invariant Mapping*, CVPR, (2006), pp. 1735–1742.

[181] R. JAIN, D. DOERMANN, *Writer Identification using an Alphabet of Contour Gradient Descriptors*, ICDAR, (2013), pp. 550–554.

[182] R. JAIN, D.S. DOERMANN, *Combining Local Features for Offline Writer Identification*, ICFHR, (2014), pp. 583–588.

[183] R. JAIN, V. FRINKEN, C. V. JAWAHAR, R. MANMATHA, *BLSTM Neural Network based Word Retrieval for Hindi Documents*, ICDAR, (2011), pp. 83–87.

[184] R. K. HANUSIAK, L. S. OLIVEIRA, E. JUSTINO, R. SABOURIN, *Writer Verification using Texture-based Features*, IJDAR, 15 (2012), pp. 213–226.

[185] R. LIKERT, *A Technique for the Measurement of Attitudes*, Archives of Psychology, New York, 140 (1932), pp. 1–55.

[186] R. PLAMONDON, G. LORETTE, *Automatic Signature Verification and Writer Identification - The State of the Art*, Pattern Recognition, 22 (1989), pp. 107–131.

[187] R. PLAMONDON, S. N. SRIHARI, *Online and Off-line Handwriting Recognition: A Comprehensive Survey*, IEEE Trans. on PAMI, 22 (2000), pp. 63–84.

[188] R. SARKAR, N. DAS, S. BASU, M. KUNDU, M. NASIPURI, D. K. BASU, *CMATERdb1: A Database of Unconstrained Handwritten Bangla and Bangla-English Mixed Script Document Image*, IJDAR, 15 (2012), pp. 71–83.

[189] S. A. MAADEED, A. HASSAINE, *Automatic Prediction of Age, Gender, and Nationality in Offline Handwriting*, EURASIP Journal on Image and Video Processing, 10 (2014), pp. 1–10.

[190] S. A. MAADEED, W. AYOUBY, A. HASSAINE, J. M. ALJAAM, *QUWI: An Arabic and English Handwriting Dataset for Offline Writer Identification*, ICFHR, (2012), pp. 746–751.

[191] S. A. MAHMOUD, I. AHMAD, W. G. A. KHATIB, M. ALSHAYEB, M. T. PARVEZ, V. MARGNER, G. A. FINK, *KHATT: An Open Arabic Offline Handwritten Text Database*, Pattern Recognition, 47 (2014), pp. 1096–1112.

[192] S. BISWAS, A. K. DAS, *Content Independent Writer Identification using Occurrences of Writing Styles for Bangla Handwritings*, Nat. Conf. on Computer Vision, Pattern Rec., Image Proc. and Graphics, (2011), pp. 154–157.

[193] ——, *Writer Identification of Bangla Handwritings by Radon Transform Projection Profile*, DAS, (2012), pp. 215–219.

[194] S. CHANDA, K. FRANKE, U. PAL, T. WAKABAYASHI, *Text Independent Writer Identification for Bengali Script*, ICPR, (2010), pp. 2005–2008.

[195] S. CHOPRA, R. HADSELL, Y. LECUN, *Learning A Similarity Metric Discriminatively, with Application to Face Verification*, CVPR, 1 (2005), pp. 539–546.

[196] S. F. BOLICH, *History of Handwriting Analysis*, Author House, Indiana, 2009, ch. I of "What America Lost: Decades that Made A Difference: Tracking Attitude Changes through Handwriting", pp. 1–4.

[197] S. FIEL, R. SABLATNIG, *Writer Identification and Writer Retrieval using the Fisher Vector on Visual Vocabularies*, ICDAR, (2013), pp. 545–549.

[198] ——, *Writer Identification and Retrieval using a Convolutional Neural Network*, Int. Conf. on Computer Analysis of Images and Patterns, (2015), pp. 26–37.

[199] S. GAZZAH, N. E. B. AMARA, *Arabic Handwriting Texture Analysis for Writer Identification using the DWT-Lifting Scheme*, ICDAR, (2007), pp. 1133–1137.

[200] S. GERTH ET AL., *Is Handwriting Performance Affected by the Writing Surface? Comparing Preschoolers', Second Graders', and Adults' Writing Performance on a Tablet vs. Paper*, Frontiers in Psychology, 7, article no. 1308 (2016).

[201] S. HAYKIN, *Neural Networks: A Comprehensive Foundation*, 2nd Eds., Prentice Hall, 1998.

[202] S. HE, L. SCHOMAKER, *Delta-n Hinge: Rotation-Invariant Features for Writer Identification*, ICPR, (2014), pp. 2023–2028.

[203] ——, *General Pattern Run-Length Transform for Writer Identification*, DAS, (2016), pp. 60–65.

[204] ——, *Writer Identification using Curvature-Free Features*, Pattern Recognition, 63 (2017), pp. 451–464.

[205] S. HE, M. WIERING, L. SCHOMAKER, *Junction Detection in Handwritten Documents and its Application to Writer Identification*, Pattern Recognition, 48 (2015), pp. 4036–4048.

[206] S. HOCHREITER, J. SCHMIDHUBER, *Long Short-Term Memory*, Neural Computation, 9 (1997), pp. 1735–1780.

[207] S. IOFFE, C. SZEGEDY, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, Int. Conf. on Machine Learning (ICML), (2015), pp. 448–456.

[208] S. J. PAN, Q. YANG, *A Survey on Transfer Learning*, IEEE Trans. on Knowledge and Data Engineering, 22 (2010), pp. 1345–1359.

[209] S. JAEGER, S. MANKE, J. REICHERT, A. WAIBEL, *Online Handwriting Recognition: The NPen++ Recognizer*, IJDAR, 3 (2001), pp. 169–180.

[210] S. L. CHOU, S. S. YU, *Sorting Qualities of Handwritten Chinese Characters for Setting Up a Research Database*, ICDAR, (1993), pp. 474–477.

[211] S. LEE, S. H. CHA, S. N. SRIHARI, *Combining Macro and Micro Features for Writer Identification*, DRR, (2002), pp. 155–166.

[212] S. LOWE, *The Complete Idiot's Guide to Handwriting Analysis*, ALPHA, 2007.

[213] S. MARINAI, B. MIOTTI, G. SODA, *Bag of Characters and SOM Clustering for Script Recognition and Writer Identification*, ICPR, (2010), pp. 2182–2185.

[214] S. MARINAI, E. MARINO, G. SODA, *Self-Organizing Maps for Clustering in Document Image Analysis*, Machine Learning in Document Analysis and Recognition, Springer, (2008), pp. 193–219.

[215] S. N. SRIHARI, M. J. BEAL, K. BANDI, V. SHAH, *A Statistical Model for Writer Verification*, ICDAR, (2005), pp. 1105–1109.

[216] S. N. SRIHARI, S. H. CHA, H. ARORA, S. LEE, *Individuality of Handwriting*, Journal of Forensic Sciences, 47 (2002), pp. 856–872.

[217] S. NAG, P. SHIVAKUMARA, Y. WU, U. PAL, T. LU, *New COLD Feature based Handwriting Analysis for Ethnicity / Nationality Identification*, ICFHR, (2018), pp. 523–527.

[218] S. PAL, A. ALAEI, U. PAL, M. BLUMENSTEIN, *Performance of an Off-Line Signature Verification Method Based on Texture Features on a Large Indic-Script Signature Dataset*, DAS, (2016), pp. 72–77.

[219] S. THADCHANAMOORTHY, N. D. KODIKARA, H. L. PREMARETNE, U. PAL, F. KIMURA, *Tamil Handwritten City Name Database Development and Recognition for Postal Automation*, ICDAR, (2013), pp. 793–797.

[220] T. KOHONEN, *Self-Organization and Associative Memory*, Springer Verlag, 1989.

[221] T. M. RATH, R. MANMATHA, *Word Image Matching using Dynamic Time Warping*, CVPR, 2 (2003), pp. 521–527.

[222] T. S. F. HAINES, O. M. AODHA, G. J. BROSTOW, *My Text in Your Handwriting*, ACM Trans. on Graphics, 35 (2016), pp. artl. 26, 18 pages.

[223] T. SU, T. ZHANG, D. GUAN, *HIT-MW Dataset for Offline Chinese Handwritten Text Recognition*, IWFHR, (2006).

[224] T. Y. LIN, Y. CUI, S. BELONGIE, J. HAYS, *Learning Deep Representations for Ground-to-Aerial Geolocalization*, CVPR, (2015), pp. 5007–5015.

[225] T. Y. ZHANG, C. Y. SUEN, *A Fast Parallel Algorithm for Thinning Digital Patterns*, Communications of the ACM, 27 (1984), pp. 236–239.

[226] U. GARAIN, T. PAQUET, *Off-Line Multi-Script Writer Identification using AR Coefficients*, ICDAR, (2009), pp. 991–995.

[227] U. PAL, A. BELAID, C. CHOISY, *Touching Numeral Segmentation using Water Reservoir Concept*, Pattern Recognition Letters, 24 (2003), pp. 261–272.

[228] U. PAL, S. DATTA, *Segmentation of Bangla Unconstrained Handwritten Text*, ICDAR, (2003), pp. 1128–1132.

[229] U. PAL, S. SINHA B. B. CHAUDHURI, *Multi-Script Line Identification from Indian Documents*, ICDAR, (2003), pp. 880–884.

[230] U. PORWAL, C. RAMAIAH, A. KUMAR, V. GOVINDARAJU, *Multiclass Learning for Writer Identification using Error-Correcting Codes*, DAS, (2014), pp. 16–20.

[231] U. V. MARTI, H. BUNKE, *Using a Statistical Language Model to Improve the Performance of an HMM-based Cursive Handwriting Recognition Systems*, IJPRAI, 15 (2001), pp. 65–90.

[232] ——, *The IAM-Database: An English Sentence Database for Off-line Handwriting Recognition*, IJDAR, 5 (2002), pp. 39–46.

[233] U. V. MARTI, R. MESSERLI, H. BUNKE, *Writer Identification using Text Line Based Features*, ICDAR, (2001), pp. 101–105.

[234] V. BOULETREAU, N. VINCENT, R. SABOURIN, H. EMPTOZ, *Synthetic Parameters for Handwriting Classification*, ICDAR, 1 (1997), pp. 102–106.

[235] V. CHRISTLEIN, D. BERNECKER, A. MAIER, E. ANGELOPOULOU, *Offline Writer Identification using Convolutional Neural Network Activation Features*, German Conference on Pattern Recognition, (2015), pp. 540–552.

[236] V. CHRISTLEIN, D. BERNECKER, E. ANGELOPOULOU, *Writer Identification using VLAD Encoded Contour-Zernike Moments*, ICDAR, (2015), pp. 906–910.

[237] V. CHRISTLEIN, D. BERNECKER, F. HONIG, A. MAIER, E. ANGELOPOULOU, *Writer Identification using GMM Supervectors and Exemplar-SVMs*, Pattern Recognition, 63 (Mar., 2017), pp. 258–267.

[238] V. FRINKEN, A. FISCHER, H. BUNKE, R. MANMATHA, *Adapting BLSTM Neural Network based Keyword Spotting Trained on Modern Data to Historical Documents*, ICFHR, (2010), pp. 352–357.

[239] V. GOVINDARAJU, S. N. SRIHARI, *Image Quality and Readability*, International Conference on Image Processing, 3 (1995), pp. 324–327.

[240] V. LAVRENKO, T. M. RATH, R. MANMATHA, *Holistic Word Recognition for Handwritten Historical Documents*, Int. Workshop on Document Image Analysis for Libraries, (2004), pp. 278–287.

[241] V. N. VAPNIK, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, (2000).

[242] V. NAIR, G. E. HINTON, *Rectified Linear Units Improve Restricted Boltzmann Machines*, Int. Conf. on Machine Learning (ICML), (2010), pp. 807–814.

[243] V. PERVOUCHINE, G. LEEDHAM, *Extraction and Analysis of Document Examiner Features from Vector Skeletons of Grapheme 'th'*, DAS, (2006), pp. 196–207.

[244] ——, *Extraction and Analysis of Forensic Document Examiner Features Used for Writer Identification*, Pattern Recognition, 40 (2007), pp. 1004–1013.

[245] W. YANG, L. JIN, M. LIU, *Chinese Character-level Writer Identification using Path Signature Feature, Dropstroke and Deep CNN*, ICDAR, (2015), pp. 546–550.

[246] X. LI, X. DING, *Writer Identification of Chinese Handwriting using Grid Microstructure Feature*, Int. Conf. on Biometrics (ICB), (2009), pp. 1230–1239.

[247] X. LI, X. DING, X. WANG, *Semi-text-independent Writer Verification of Chinese Handwriting*, ICFHR, (2008), p. # 1021.

[248] X. WU, Y. TANG, W. BU, *Offline Text-Independent Writer Identification Based on Scale Invariant Feature Transform*, IEEE Trans. on Information Forensics and Security, 9 (2014), pp. 526–536.

[249] X. X. NIU, C. Y. SUEN, *A Novel Hybrid CNN-SVM Classifier for Recognizing Handwritten Digits*, Pattern Recognition, 45 (2012), pp. 1318–1325.

[250] Y. AKAO, A. YAMAMOTO, Y. HIGASHIKAWA, *Assisting Forensic Writer Verification by Visualizing Diversity of Digit Handwritings: An Approach by Multidimensional Scaling of Earth Mover's Distance*, ICFHR, (2014), pp. 110–115.

[251] Y. AKBARI, K. NOURI, J. SADRI, C. DJEDDI, I. SADDIQI, *Wavelet-based Gender Detection on Offline Handwritten Documents using Probabilistic Finite State Automata*, Image and Vision Computing, (2017), pp. 17–30.

[252] Y. J. XIONG, Y. LU, P. S. P. WANG, *Off-line Text-Independent Writer Recognition: A Survey*, IJPRAI, 31 (2017), p. #1756008 (32 pages).

[253] Y. J. XIONG, Y. WEN, P. S. P. WANG, Y. LU, *Text-Independent Writer Identification using SIFT Descriptor and Contour-Directional Feature*, ICDAR, (2015), pp. 91–95.

[254] Y. LECUN, L. BOTTOU, Y. BENGIO, P. HAFFNER, *Gradient-based Learning Applied to Document Recognition*, Proceedings of the IEEE, 86 (1998), pp. 2278–2324.

[255] Y. MA, G. GUO, *Support Vector Machines Applications*, Springer International Publishing, Switzerland, (2014).

[256] Y. TANG, X. WU, *Text-independent Writer Identification via CNN Features and Joint Bayesian*, ICFHR, (2016), pp. 566–571.

[257] Y. ZHU, T. TAN, Y. WANG, *Biometric Personal Identification based on Handwriting*, ICPR, 2 (2000), pp. 797–800.

[258] Z. A. DANIELS, H. S. BAIRD, *Discriminating Features for Writer Identification*, ICDAR, (2013), pp. 1385–1389.

[259] Z. HE, X. YOU, Y. Y. TANG, *Writer Identification of Chinese Handwriting Documents using Hidden Markov Tree Model*, Pattern Recognition, 41 (2008), pp. 1295–1307.

[260] ——, *Writer Identification using Global Wavelet-based Features*, Neurocomputing, 71 (2008), pp. 1832–1841.

[261] Z. HE, Y. Y. TANG, B. FANG, J. DU, X. YOU, *A Novel Method for Off-line Handwriting-based Writer Identification*, ICDAR, (2005), pp. 242–246.