



Faculty of Computer Science and Information Technology

**Contour-KNN Brahmi Segmentation (CKBS) and Two-Phase Enhanced
Brahmi Recognition (TREBR) Methods for Automatic Brahmi Texts
Labelling**

Neha Gautam

**Doctor of Philosophy
2024**

Contour-KNN Brahmi Segmentation (CKBS) and Two-Phase Enhanced
Brahmi Recognition (TREBR) Methods for Automatic Brahmi Texts
Labelling

Neha Gautam

A thesis submitted

In fulfillment of the requirements for the degree of Doctor of Philosophy

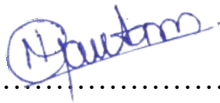
(Computer Science)

Faculty of Computer Science and Information Technology
UNIVERSITI MALAYSIA SARAWAK

2024

DECLARATION

I declare that the work in this thesis was carried out in accordance with the regulations of Universiti Malaysia Sarawak. Except where due acknowledgements have been made, the work is that of the author alone. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.



.....
Signature

Name: Neha Gautam

Matric No.: 16010146

Faculty of Computer Science and Information Technology

Universiti Malaysia Sarawak

Date: 20 Jan 2024

ACKNOWLEDGEMENT

The time spent completing my PhD has been without question one of the most challenging and rewarding experiences of my life. I have amassed a wealth of knowledge and learned much about myself. I am certain that I would not have been able to complete this journey without the support of several individuals. It is with humble gratitude that I would like to recognise them here.

First and foremost, I would like to praise and thank God, the Almighty, who has granted countless blessings, knowledge, and opportunities to the writer so that I have finally been able to accomplish the thesis.

I express my deep gratitude to my supervisor, Dr. Chai Soo See, for guiding me well throughout the research work, from title selection to finding the results. The immense knowledge, motivation, and patience of Dr. Chai have given me more power and spirit to excel in research writing. Conducting the academic study regarding such a difficult topic could not be as simple as she made this for me.

I am also indebted to the Zamalah Siswazah UNIMAS Scholarship (ZSU) for generously providing the financial support needed to pursue my study. Without their assistance, I would not have been able to progress as far as I have.

I would never forget my fellow labmates for the fun time we spent together, the sleepless nights that gave us the courage to complete tasks before deadlines, and for stimulating the discussions.

Ultimately, I am grateful to my parents, siblings, friends, and acquaintances who remembered me in their prayers for the ultimate success. I consider myself nothing without them. They gave me enough moral support, encouragement, and motivation to accomplish my goals. This thesis is dedicated to my father.

ABSTRACT

Automatic word recognition problem can be solved using an optical character recognition (OCR) system. Few studies have been seen in the field of Brahmi word recognition especially identifying compound characters and words with good accuracy. However, existing Brahmi text recognition studies have primarily relied on local datasets, hampering the standardization of datasets. To address this, the study proposes a systematic dataset creation process that encompasses data collection, pre-processing, segmentation, data augmentation, recognition, storage, and labelling. The process is initiated with data collection from diverse online sources, yielding 217 text and word samples and 801 isolated characters and compound characters. However, these samples lack uniformity in text and word sizes. The subsequent phase focuses on character isolation from words and text, utilizing a novel segmentation approach as a crucial precursor to system training. A Contour-KNN Brahmi Segmentation (CKBS) for character and compound character segmentation is introduced. Object detection identifies characters, including dots (.), and links them to their nearest left character using KNN. This approach greatly enhances segmentation, achieving an impressive 98.19% average accuracy. The segmentation approach generates 40 samples per class across 170 classes, with a 75:25 training-testing split (30 and 10 samples for training and testing, respectively). Furthermore, data augmentation techniques, including adjustments, deformations, blurring, translations, and noise introduction, are applied to enhance dataset quality and quantity. Data augmentation results in 180 training and 60 testing samples per class, improving both size and quality. Subsequently, a Two-Phase Enhanced Brahmi Recognition (TPEBR) is employed, distinguishing between global and local feature recognition. Various deep learning architectures are evaluated for classification, with resizing to meet specific input size requirements. SqueezeNet emerges as the most

effective, achieving a minimal 0.237 loss and an exceptional 97.58% accuracy. It excels in precision, recall, and F1-score. In contrast, ResNeXt Small underperforms with higher loss and lower accuracy. Comparing the Two-Phase Enhanced Brahmi Recognition (TPEBR) to the existing approaches, the Two-Phase Enhanced Brahmi Recognition (TPEBR) achieves 97.58% accuracy, while the existing approaches records 80.20% and 90.24%. Recognized characters are then organized into folders according to their recognized class, and done labelling by using Brahmi Unicode, although this step does not impact performance of the system.

Keywords: Brahmi script, deep learning architecture, data augmentation, KNN, object detection, two-phase approach, word recognition

Segmentasi Brahmi KNN Kontur (CKBS) dan Kaedah Pengiktirafan Brahmi Dua Fasa yang Dipertingkatkan (TREBR) untuk Pelabelan Teks Brahmi Automatik

ABSTRAK

Masalah pengenalan perkataan secara automatik boleh diselesaikan dengan menggunakan sistem pengenalan aksara optik (OCR). Beberapa kajian telah dilihat dalam bidang pengenalan perkataan Brahmi terutamanya mengenal pasti aksara gabungan dan perkataan dengan ketepatan yang baik. Walau bagaimanapun, kajian pengenalan teks Brahmi yang sedia ada kebanyakannya bergantung pada set data tempatan, menghalang pilaian set data. Untuk menangani ini, kajian ini mencadangkan proses penciptaan set data yang sistematik yang merangkumi pengumpulan data, pra-pemprosesan, segmentasi, penambahbaikan data, pengenalan, penyimpanan, dan pelabelan. Proses ini dimulakan dengan pengumpulan data dari pelbagai sumber dalam talian, menghasilkan 217 sampel teks dan perkataan dan 801 aksara terasing dan aksara gabungan. Namun, sampel ini kekurangan keseragaman dalam saiz teks dan perkataan. Fasa seterusnya memberi tumpuan kepada pengasingan aksara dari perkataan dan teks, menggunakan pendekatan segmentasi baru sebagai pendahuluan penting kepada latihan sistem. Satu Contour-KNN Brahmi Segmentation (CKBS) untuk segmentasi aksara dan aksara gabungan telah diperkenalkan. Pengesanan objek mengenal pasti aksara, termasuk titik (.), dan mengaitkannya dengan aksara kiri terdekat menggunakan KNN. Pendekatan ini meningkatkan segmentasi dengan ketara, mencapai ketepatan purata 98.19%. Pendekatan segmentasi menghasilkan 40 sampel setiap kelas di 170 kelas, dengan bahagian latihan-uji 75:25 (30 dan 10 sampel untuk latihan dan ujian, masing-masing). Selain itu, teknik penambahbaikan data, termasuk penyesuaian, deformasi, kabur, terjemahan, dan pengenalan bising, diterapkan untuk meningkatkan kualiti dan kuantiti set data. Penambahbaikan data menghasilkan 180 sampel

latihan dan 60 sampel ujian setiap kelas, meningkatkan saiz dan kualiti. Kemudian, kaedah *Two-Phase Enhanced Brahmi Recognition (TPEBR)* yang diperkenalkan dalam kajian ini telah digunakan untuk membezakan antara pengenalan ciri global dan tempatan. Pelbagai seni bina pembelajaran mendalam dinilai untuk pengelasan, dengan mengubah saiz untuk memenuhi keperluan saiz input tertentu. *SqueezeNet* muncul sebagai yang paling berkesan, mencapai kerugian minimum 0.237 dan ketepatan luar biasa 97.58%. Ia cemerlang dalam ketepatan, ingatan, dan skor *F1*. Sebagai perbandingan, *ResNeXt Small* kurang berprestasi dengan kerugian yang lebih tinggi dan ketepatan yang lebih rendah. Membandingkan *Two-Phase Enhanced Brahmi Recognition (TPEBR)* dengan pendekatan yang sedia ada, *Two-Phase Enhanced Brahmi Recognition (TPEBR)* mencapai ketepatan 97.58%, manakala pendekatan yang sedia ada mencatat 80.20% dan 90.24%. Aksara yang dikenali kemudian dianjurkan ke dalam folder mengikut kelas yang dikenali, dan dilakukan pelabelan dengan menggunakan *Unicode Brahmi*, walaupun langkah ini tidak memberi kesan kepada prestasi sistem.

Kata kunci: Skrip brahmi, seni bina pembelajaran mendalam, penambahbaikan data, KNN, pengesanan objek, pendekatan dua fasa, pengenalan perkataan.

TABLE OF CONTENTS

	Page
DECLARATION	i
ACKNOWLEDGEMENT	ii
ABSTRACT	iii
<i>ABSTRAK</i>	v
TABLE OF CONTENTS	vii
LIST OF TABLES	xi
LIST OF FIGURES	xiii
CHAPTER 1 INTRODUCTION	1
1.1 Background Study	1
1.1.1. Online Text Recognition	3
1.1.2. Offline Text Recognition	3
1.2 Problem Statement	6
1.3 Research Questions	8
1.4 Objectives	9
1.5 Hypothesis	10
1.6 Scope	11
1.7 Research Methodology	12
1.8 Organization of the Thesis	13

CHAPTER 2 WRITING SYSTEMS AND SCRIPTS OF THE WORLD	16
2.1 The Writing System of the World	16
2.1.1 Logographic System	16
2.1.2 Syllabic System	16
2.1.3 Abjad System	16
2.1.4 Abugida System	17
2.1.5 True Alphabetic	18
2.1.6 Featural System	18
2.2 Overview of the Brahmi Script	20
2.2.1 Properties of Brahmi Script	20
2.2.2 Feature of Brahmi Words	22
2.3 Summary	25
CHAPTER 3 LITERATURE REVIEW	26
3.1 Brahmi Characters and Words Recognition System	26
3.2 Dataset	28
3.3 Segmentation	32
3.4 Approaches for Brahmi Text Recognition: An Overview	37
3.4.1 Holistic and Analytic Approaches	37
3.4.2 Single-phase and two-phase approach	40
3.5 Summary	45

CHAPTER 4 METHODOLOGY	46
4.1 Data Collection	46
4.2 Pre-processing	48
4.2.1 Binarization/Thresholding	48
4.2.2 Noise Removal	51
4.3 Contour-KNN Brahmi Segmentation (CKBS) Method	53
4.3.1 Object Detection	53
4.3.2 Steps of KNN	58
4.4 Data Augmentation	60
4.5 Two-Phase Enhanced Brahmi Recognition (TPEBR) Method	64
4.6 Data Storage and Labelling	70
4.7 Summary	73
CHAPTER 5 EXPERIMENTAL RESULTS	74
5.1 Result Contour-KNN Brahmi Segmentation (CKBS) (First Part)	76
5.2 Data Augmentation	78
5.3 Result of Two-Phase Enhanced Brahmi Recognition (TPEBR) Method (Second part)	78
5.3.1 Performance Metrics	79
5.3.2 Result of the Recognition Model	80
5.3.3 Comparison the performance of Two-Phase Enhanced Brahmi Recognition (TPEBR) Method with Previous Studies	90

5.4	Overall Performance of Proposed System	91
5.5	Statistical Distribution	91
5.6	Summary	92
CHAPTER 6 CONCLUSION AND RECOMMENDATION		94
6.1	Contributions	95
6.2	Limitations	97
6.2.1	Dataset	98
6.2.2	Segmentation	98
6.2.3	Brahmi Text Recognition System	98
6.3	Future Works	98
6.3.1	Dataset	98
6.3.2	Segmentation	99
6.3.3	Brahmi Text Recognition System	99
REFERENCES		100
APPENDICES		115

LIST OF TABLES

	Page
Table 2.1 Example of Compound Characters	24
Table 3.1 Comparison of the Three Approaches in Terms of Brahmi Script	39
Table 4.1 Summary of the Threshold Value is Determined Through Otsu's Thresholding Method for all of the Images	50
Table 4.2 Analysis of data augmentation techniques	61
Table 4.3 Statistics Value of Q_{11} , Q_{21} , Q_{12} , and Q_{22}	69
Table 4.4 Standard Size of each Architecture	70
Table 5.1 Performance of TPEBR Approach where Acc, Pre, Rec, F1 Represents Accuracy, Precision, Recall and F1-Score	81
Table 5.2 Performance of TPEBR Approach on Adjusted Data Augmentation Testing Sample Where Acc Represents the Accuracy	82
Table 5.3 Performance of TPEBR Approach on Blurred Data Augmentation Testing Sample where Acc Represents the Accuracy	83
Table 5.4 Performance of TPEBR Approach Approach on Deformed Data augmentation Testing Sample	84
Table 5.5 Performance of TPEBR Approach on Noisy Data Augmentation Testing Sample	85
Table 5.6 Performance of TPEBR Approach on Translated Data Augmentation Testing Sample	86
Table 5.7 Performance of TPEBR Approach on Original Data Testing Sample	87
Table 5.8 Comparison of Training and Testing Accuracy of TPEBR Approach where Acc represnts Accuracy	88
Table 5.9 The performance TPEBR Approaches of the Testing Dataset of Segmentation Approach	89

Table 5.10	Comparison of Existing and Proposed Brahmi Word Recognition Systems	90
------------	---	----

LIST OF FIGURES

	Page
Figure 1.1 Flow Diagram of the Brahmi Text Recognition System	6
Figure 1.2 Connection between Problem Statements and Research Questions	9
Figure 1.3 Connection between Research Questions and Research Objectives	10
Figure 1.4 Connection between Research Objective and Hypothesis	11
Figure 2.1 The Writing Systems of World (Ghosh et al., 2010; Obaidullah et al., 2018)	19
Figure 2.2 Characters and Vowels of the Brahmi Script (Roy & Mandal, 2016)	20
Figure 2.3 Vowels of the Brahmi Script (Roy & Mandal, 2016)	21
Figure 2.4 Example of Consonant and Compound Characters (Roy & Mandal, 2016)	21
Figure 2.5 Example Text in Brahmi Script (Rajan, 2010)	21
Figure 2.6 Example of Complex Compound Characters	22
Figure 4.1 A Diagram of the Overall Methodology	47
Figure 4.2 Samples of Brahmi Text, With and Without Noise	48
Figure 4.3 Input image	51
Figure 4.4 Output image of Erosion structuring element	52
Figure 4.5 Output image of Dilation structuring element	52
Figure 4.6 Input image	56
Figure 4.7 Output of object detection part	56
Figure 4.8 After merging the dot character with the relevant character	58
Figure 4.9 Example of all segmented character	59
Figure 4.10 Hierarchy of the Data Augmentation	60
Figure 4.11 Consonant, Vowels, and Compound Characters of the Brahmi Script	65

Figure 4.12	A Schematic Representation of Proposed Approach Two-Phase Enhanced Brahmi Recognition (TPEBR)	66
Figure 4.13	Process of Proposed Approach (Two-Phase Enhanced Brahmi Recognition (TPEBR)) with an Example	67
Figure 4.14	All Isolate Characters are Stored into Groups by using Segmentation and Recognition Step	70
Figure 4.15	Labeling of few Samples of Brahmi Dataset	71
Figure 4.16	Flow of Methodology of Proposed System to do Automatic Labelling along with Training and Testing Details	72
Figure 5.1	Flow of the Performance Evaluation of the Proposed System	75

CHAPTER 1

INTRODUCTION

This section provides an outline to develop an automated system to generate a Brahmi dataset, with a specific focus on the importance of automated word labelling in the context of the Brahmi script. It delves into the exploration of an appropriate methodology for future investigations. Furthermore, it addresses the formulation of problem statements and the establishment of research objectives. The structure of this chapter encompasses five key segments. The initial segment (1.1) furnishes the historical backdrop of the study, while the subsequent portions (1.2, 1.3, 1.4, and 1.5) elucidate the problem statements, scope of the study, research questions, and objectives, respectively. Additionally, the development of a hypothesis is presented in Section 1.6. The chapter concludes by delving into the research methodology in Section 1.7 and providing an overview of the thesis organization in Section 1.8.

1.1 Background Study

The modern world is becoming more digitised with the prevalence of computers, PDA screens, and devices, which are traditional books and newspapers. Carrying around large amounts of paper will require more maintenance because the content of the paper can disappear with time and under different weather conditions, making it tough to restore the removed content. Therefore, books, newspapers, or any sort of paper are now being converted into digital format via a scanning process to store them for a long time. However, scanned images are not automatically readable and editable by computers, so the text in the image cannot be used directly for further work. This issue can be solved using suitable recognition systems, which are capable of reading the content similar to the human ability

to read. Up until now, various types of recognition systems have been developed for automatic word recognition.

Numerous works in the field of English and Roman script recognition have been done compared to South and Central Asian scripts because South and Central Asian scripts are more cursive compared to English and Roman scripts (Gautam et al., 2016; Roy et al., 2016). Ancient script recognition of South and Central Asian scripts has brought a new challenge because of the significant variations in the structure of the characters, lack of resources, controversy of statements, etc (Sulaiman et al., 2019).

The Brahmi script was an ancient script that was once used in South Asia, Southeast Asia, and East Asia. A study identified 198 different contemporary scripts, such as Devanagari, Bangla, Gurmukhi, Gujarati, Oriya, etc., all descended from the Brahmi script (Gautam & Chai, 2017; Acharya, 2018). Hence, it can be said that the Brahmi script has a rich background. The Brahmi script has its origins in Theravada Buddhism, which is known as the “pillars of Asoka”, where the text on the pillars is written in Brahmi (Gautam & Chai, 2017). Similarly, text related to the Buddhist culture is also written using the Brahmi script, mainly appearing on various pillars, caves, and stones. These pillars, caves, and stones were the main ways of spreading Buddhist theology all over Asia, mostly in South Asia, Southeast Asia, and East Asia (Igunma, 2018).

Some studies in the realm of Brahmi text recognition have delivered a good performance. These investigations initially focused on the extraction of pertinent features from Brahmi text. Subsequently, the extracted features were subjected to classification via suitable algorithms. Diverse methods for feature extraction and classification were employed. For instance, Geometric feature extraction (Gautam & Chai, 2017) and zoning techniques (Gautam et al., 2016; Vellingiriraj et al., 2016) were utilized to acquire features

of Brahmi characters and Brahmi text. In contrast, a combination of rule-based systems (Gautam & Chai, 2017), template matching (Gautam et al., 2016), and Artificial Neural Networks (ANN) (Vellingiriraj et al., 2016) served as classifiers to categorize the extracted features.

The challenge of Brahmi text recognition falls within the broader domain of automatic word recognition, which can be primarily categorized as online and offline text recognition. When data is captured from paper via optical scanners or cameras, this process is known as offline text recognition. Meanwhile, a system that uses digitisers to capture written content at the time of writing is called online text recognition.

1.1.1 Online Text Recognition

In this process, words are recognised in real-time as soon as the content is written, and thus, it incorporates temporal information. Online systems have replaced the role of the pen as a function of time directly from the interface. The online recognition system is usually done through pen-based interfaces where the writer writes with a special pen on an electronic tablet.

1.1.2 Offline Text Recognition

Offline text recognition systems follow a particular process. Firstly, the text must be written on paper. The paper is then scanned with a scanner or a camera. Following the capture and scanning of the text in picture format, optical character recognition (OCR) is used to transform the image into a readable and editable format (Das et al., 2019; Sharma & Mudgal, 2019). OCR is a technology that allows for the conversion of a variety of documents, such as scanned paper documents and PDF files or pictures, into editable and searchable data. (Chaudhuri et al., 2017; Budhi et al., 2018).

The Brahmi script is an old script with a lot of written information. In the world of today, this script is no longer used for any sort of communication. Hence, offline text recognition is the most useful solution for recognising Brahmi script, with OCR employed to locate Brahmi text. With offline word recognition, analytical and holistic approaches are used to develop the Brahmi text recognition system.

1.1.2.1 Analytical Process

In the analytical process, a word is initially segmented into sub-units called characters, and then each sub-unit is categorised into classes. This step helps to train and test the recognition system. Most word recognition solutions were developed based on the analytical process.

1.1.2.2 Holistic Process

The holistic approach considers a word as a single and unified unit. Thus, the text is only segmented into meaningful words, and all words are categorised into classes. This step will help to build the system. This approach addresses the problems of the fixed or limited lexicon. Besides that, this approach can be used to reduce the lexicon in large vocabulary problems (Madhvanath et al., 1999; Madhvanath & Govindaraju, 2001).

In the comparison of both approaches, the analytical process is useful with ancient scripts where not a lot of text is available of the respective script. However, the major problem with the analytical process is that all characters are separately recognised, so word prediction might not be done because word predictions are only possible if the whole word has been recognised. Word prediction is used to increase the accuracy of the recognition system.

The holistic approach can perform well in popular scripts such as English, Arabic, and Chinese as these are the modern scripts and many dictionaries are available which can help to do word prediction.

For Brahmi text recognition, available samples of Brahmi words and text are limited, so the holistic approach might not be more suitable than the analytical approach for a Brahmi text recognition system. The holistic system also requires the use of a dictionary or book that can help to provide information about all possible words or vocabularies. However, from the review conducted in this study, no official dictionary for Brahmi words and no such book that could help to collect information regarding the meaning of the words in the Brahmi script could be found. Therefore, OCR based on the analytical process is more suitable for building the Brahmi text recognition system in this study.

Input, pre-processing, segmentation, feature extraction, and classification are the main steps employed by the OCR system. Pre-processing enhances the quality of the input image. The segmentation step isolates the characters from the text and word and is optional. In scenarios where individual characters are already separated, the segmentation process becomes unnecessary. However, for systems focusing on text recognition, this step gains significance. It follows a sequence where segmentation starts from the line, then progresses to the word level, and finally culminates with the segmentation of individual characters. Furthermore, feature extraction and classification are the decision-making steps of the OCR. Two types of paradigms can be followed to complete the feature extraction and classification steps for offline Brahmi words: supervised and unsupervised. Supervised learning identifies an unknown pattern as a member of the predefined category, while unsupervised learning groups input patterns into several clusters - hereafter defined as classes. Various methods are used to extract the features of each class, and some classifiers are also used to classify

the extracted features in the supervised learning method. In contrast, in supervised learning, there are no explicit target outputs or environmental evaluations associated with each input; rather, previous biases about which features of the input structure should be represented in the output are brought to bear in unsupervised learning.

The methods for developing a Brahmi text recognition system are illustrated in Figure 1.1.

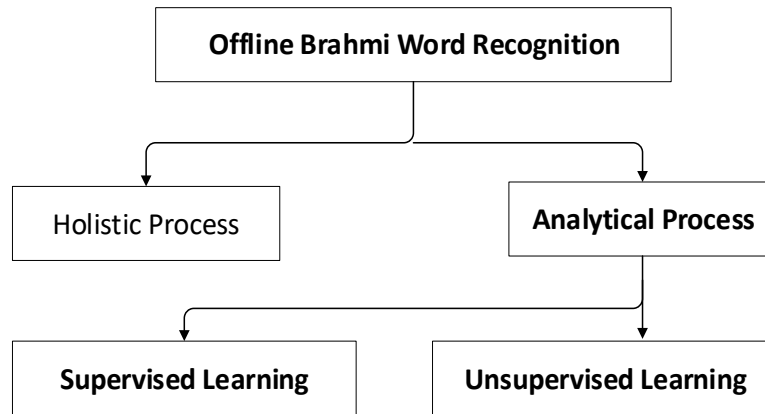


Figure 1.1: Flow Diagram of the Brahmi Text Recognition System

1.2 Problem Statement

The dataset of a study has always played an important role in word recognition systems, as it helps to train the model and if the dataset is publicly available, it provides a transparent platform for comparing the results of one study to that of other studies. In 2016, Vellingiriraj et al. (2016), who worked with Brahmi text recognition, also noticed that an official dataset would be required for Brahmi script recognition to evaluate the performance of the Brahmi script recognition algorithms. Currently, researchers investigating Brahmi script recognition have used local datasets with limited samples, which are unsuitable for further studies as such datasets are not available in the public domain. Accordingly, a different Brahmi script dataset was used in each study. A publicly available dataset should

be used as a transparent platform for a more reliable comparison of the different recognition algorithms. Thus, such studies require a Brahmi word dataset.

Furthermore, as previously emphasized, an analytical approach stands as a more viable choice for Brahmi text recognition, primarily due to the significance of a comprehensive dictionary. Consequently, the dataset must meticulously organize segmented characters within dedicated directories. Such a dataset holds the potential for advancing recognition tasks in subsequent stages. Nevertheless, the Brahmi script being an ancient writing system, no longer serves as a means of communication in the contemporary era. Therefore, automating both tasks could offer researchers valuable support in the times ahead. For the automated labelling of individual characters, the initial step involves character recognition. This signifies that the complete workflow, spanning from data collection and organized storage in the relevant directory, commences with pre-processing, segmentation, recognition of Brahmi words, and culminates in the process of automatic labelling.

Numerous segmentation methods have been developed for scripts, such as English (Kaur & Singh, 2016), Arabic (Ghaleb et al., 2017), and Devanagari (Jindal & Kumar, 2018), which aid in the development of recognition systems. Similarly, few techniques have been proposed for isolating Brahmi characters from Brahmi text. However, these studies have not been successful in segmenting all types of characters. For instance, Gautam et al. (2016) were unable to isolate characters with dot (.) features. Additionally, Singh and Kushwaha (2019) recently introduced a Brahmi text segmentation algorithm, but it did not achieve satisfactory performance due to the unique features of the Brahmi script. As a result, this remains an open research problem (Singh & Kushwaha, 2019).

Furthermore, an automated Brahmi word recognition system can be helpful for understanding each word of the Brahmi text. Machine learning can be useful to achieve such

an automated system (Dargan et al., 2020). Existing works that have developed Brahmi script recognition systems and other recognition systems should serve as a foundation for new investigations into the recognition of Brahmi script. Various researchers such as Siromoney et al. (1983), Soumya and Kumar (2015), Vellingiriraj et al. (2016) have been working to recognize the Brahmi word. However, Vellingiriraj et al. (2016) mentioned that the performance of the existing Brahmi word recognition may be further improved.

Moreover, as previously emphasized, the Brahmi script is understood by very few researchers. Therefore, automating the labelling of the dataset can be a valuable step to assist future researchers in improving both the quality and quantity of the dataset. Hence, after recognizing Brahmi characters and compound characters, automatically organizing them into a dictionary and applying automatic labels to each entry becomes an essential task that has not been addressed in the context of the Brahmi script.

1.3 Research Questions

Based on the issues presented in the problem statement, several research inquiries emerge, with a specific focus on aspects such as the dataset, segmentation, and recognition procedures. These research questions are denoted by the abbreviation "RQ" signifying their prominence in the study.

RQ1: How can a dataset be effectively developed for the Brahmi text recognition system?

RQ2: What automated procedures can be implemented for labelling and storing data within the Brahmi text recognition dataset?

RQ3: What approach can be devised to efficiently segment various character categories present within the Brahmi text?

RQ4: How can an automatic Brahmi text recognition system be conceptualized and implemented?

The relationship between the problem statements and research questions is presented in Figure 1.2.

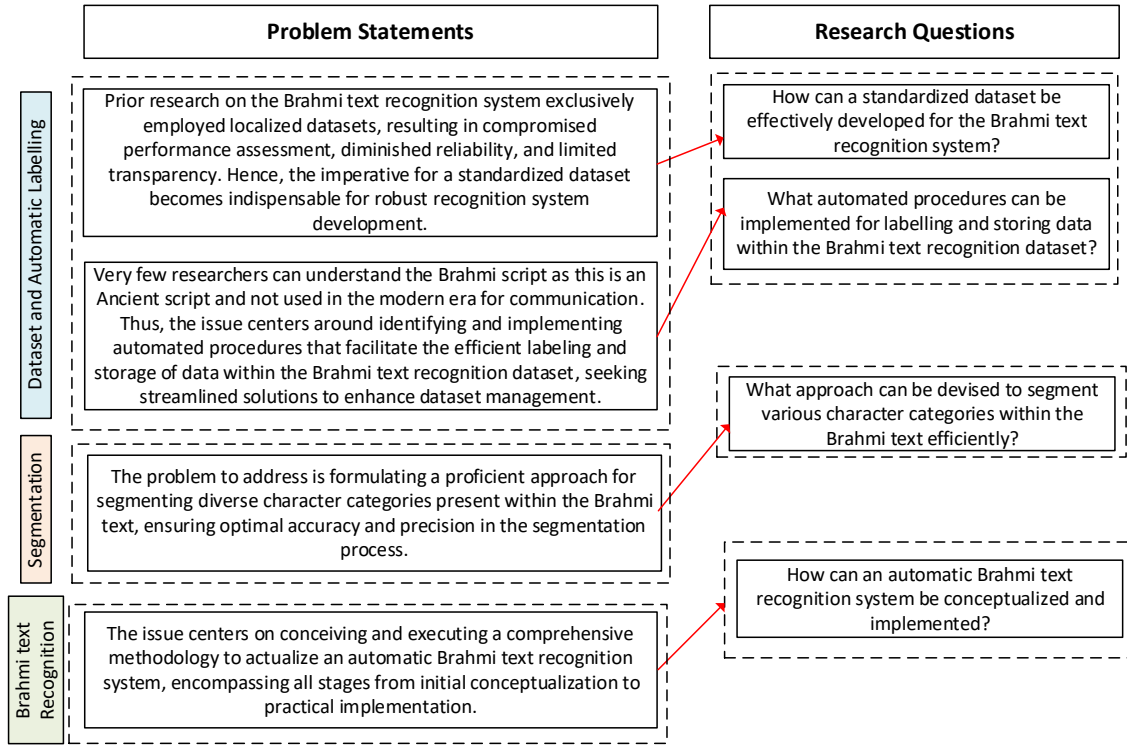


Figure 1.2: Connection between Problem Statements and Research Questions

1.4 Objectives

The central aim of this research is to investigate and create a mechanism facilitating the automatic labelling of Brahmi words, followed by their organized arrangement into designated folders. This overarching objective seeks to enhance the effectiveness of segmentation and recognition, building upon the principles previously outlined by the segmentation and recognition system of Brahmi script. In pursuit of this primary goal, the research aligns with the following specific Research Objectives (RO):

RO1: To establish a dataset incorporating automated processes encompassing labelling and storage.

RO2: To introduce and implement an approach capable of proficiently segmenting diverse characters present within Brahmi text.

RO3: To introduce and develop a procedure for an automatic Brahmi text recognition system.

The relationship between the research questions and research objectives is presented in Figure 1.3.

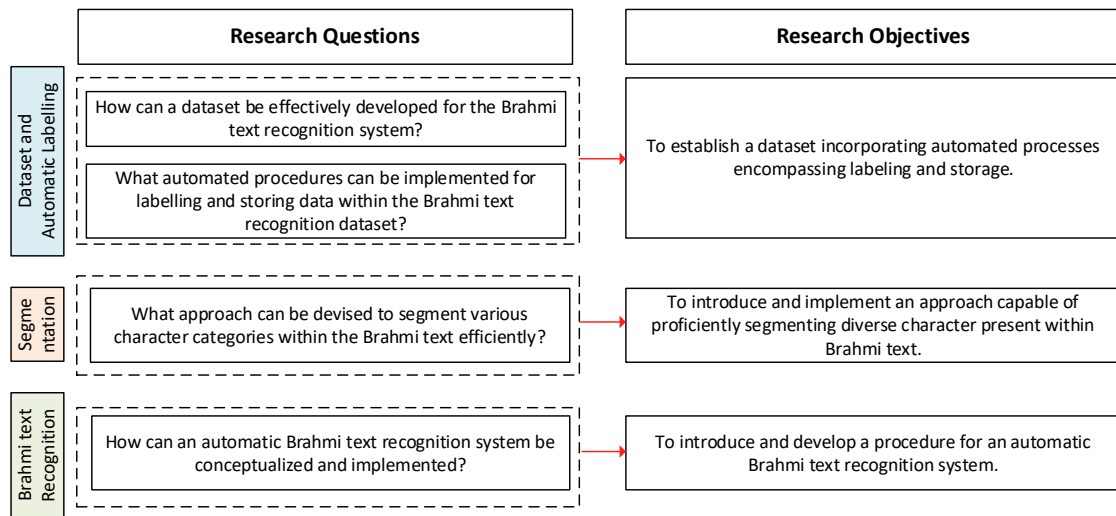


Figure 1.3: Connection between Research Questions and Research Objectives

1.5 Hypothesis

From the considerations laid out in the research objectives, multiple hypotheses come forth, concentrating particularly on facets encompassing the dataset, segmentation techniques, and recognition processes. These hypotheses are identified by the abbreviation "H" symbolizing their paramount importance within the study.

H1. The automated processes are implemented to create a dataset, encompassing labelling and storage for the Brahmi text recognition system.

H2. The establishment and proficient execution of a method adept at segmenting varied character categories within the Brahmi text will yield dependable and accurate outcomes in the realm of segmentation.

H3. The proposed approach for an automatic Brahmi text recognition system will increase the consistent and accurate results for data that are entirely independent from the training dataset.

The relationship between the research objective and hypothesis is presented in Figure 1.4.

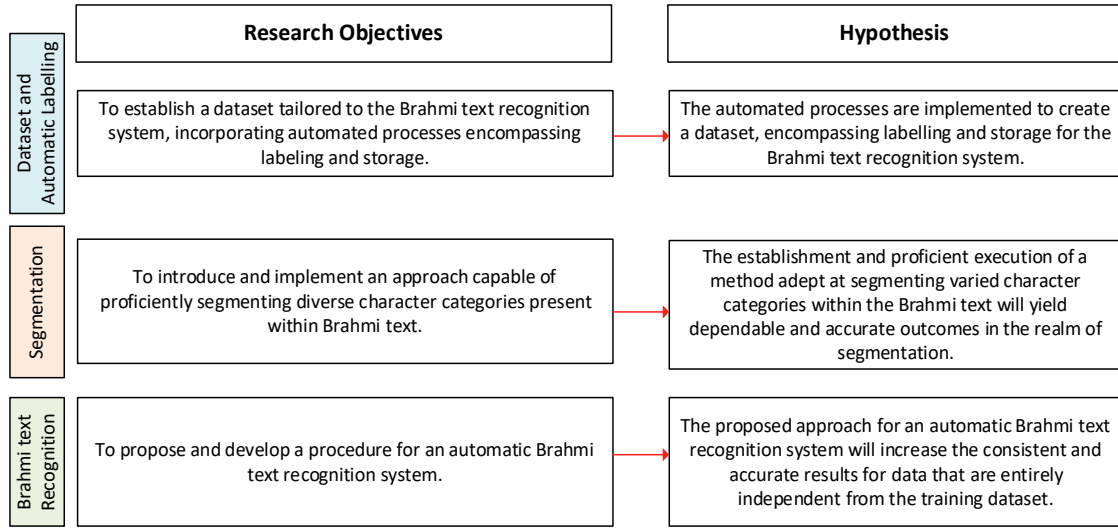


Figure 1.4: Connection between Research Objective and Hypothesis

1.6 Scope

This research project aims to explore and address key challenges in the domain of Brahmi text recognition. The scope encompasses the development and implementation of automated solutions for character labelling, segmentation, and recognition within the context of the Brahmi script. The research seeks to establish a comprehensive understanding of the intricacies involved in creating a dataset, devising efficient segmentation methodologies, and conceptualizing an automatic recognition system.

The scope of this research extends to investigating various aspects, including dataset creation, automated labelling and storage procedures, advanced segmentation techniques, cross-script applicability, and the development of an automatic recognition mechanism.

Furthermore, the research will conduct a critical review of the existing literature related to Brahmi text recognition, offering a comprehensive overview of the current state and potential areas of the field for improvement. The research scope emphasizes the practical applicability of the proposed solutions and their potential impact on enhancing the accuracy and efficiency of Brahmi text recognition systems.

Overall, this research project seeks to contribute to the advancement of Brahmi text recognition technology by addressing fundamental challenges and exploring innovative approaches that have the potential to significantly improve the segmentation and recognition processes within the Brahmi script.

1.7 Research Methodology

The methodology encompasses the formulation of strategies to accomplish the objectives outlined in Section 1.4. This study is structured into four distinct sections, each addressing specific aspects. The initial segment is dedicated to the collection of printed Brahmi word samples. Subsequently, the focus shifts to the pre-processing and segmentation phases. Pre-processing is geared toward optimizing image quality according to specific criteria, while segmentation involves the comprehensive extraction of various character types, encompassing compound characters, from the Brahmi text. The effectiveness of the proposed segmentation technique is subsequently gauged by assessing its performance on both a specialized Brahmi dataset and a comparable dataset possessing script characteristics akin to Brahmi.

In the third segment, data augmentation techniques (adjusted, deformed, blurred, translated, and noisy) are applied to all segmented characters and compound characters to enhance the quantity and quality of the samples. Furthermore, the study hones in on enhancing the recognition rate of the Brahmi text recognition system. This stage involves

the division of the Brahmi dataset into distinct training and testing sets. The developed process is then subjected to evaluation using the Brahmi dataset to validate the accuracy of model. Lastly, under the automatic labelling section, all samples are automatically organized into appropriate folders based on character features.

1.8 Organization of the Thesis

This thesis consists of seven chapters for which the summary is discussed below.

CHAPTER 2 provides information about the origin, writing style, and characteristics of various scripts of the writing systems of world. The characteristics of the different writing systems and the structural properties of the characters are also explained. The Brahmi script is a part of this writing system, with origins in the Brahmic family. The characteristics, properties and features of the compound characters in the Brahmi script are discussed.

CHAPTER 3 provides an insightful exploration of studies pertaining to diverse recognition system datasets. The discussion centres on identifying research gaps and identifying a suitable methodology for constructing datasets, performing segmentation, and advancing Brahmi text recognition. The literature examination commences by scrutinizing a range of studies focused on script segmentation worldwide. This scrutiny facilitates the identification of an optimal approach tailored to the distinctive features of Brahmi characters, thereby enhancing segmentation accuracy and overcoming limitations inherent in existing Brahmi text segmentation techniques.

Additionally, the review encompasses an assessment of existing recognition systems to inform the process employed in this study - a two-phase strategy employing multiple classifiers to further heighten the accuracy of Brahmi text recognition. Furthermore, a thorough examination of Brahmi character and word recognition system studies is conducted, aiming to ascertain appropriate techniques for training and testing each classifier

within the proposed approach. The cumulative outcome of this chapter significantly enhances the grasp of researcher of the methodology, dataset techniques, segmentation procedures, and the intricate mechanics underpinning the Brahmi word recognition system.

CHAPTER 4 presents the methodology for constructing the dataset underpinning the Brahmi text recognition system is meticulously presented. This method entails a series of sequential steps, encompassing data collection, pre-processing, segmentation, data augmentation, recognition, storage, labelling, and the establishment of a statistical distribution, including details regarding the number of samples.

This chapter expounds upon the proposed technique for character segmentation within Brahmi words. Notably, the efficacy of the segmentation approach is scrutinized using an unseen dataset, entirely distinct from the training dataset. Moreover, the performance of segmentation technique is also tested on Tamil script text, notable for its shared characteristics with the Brahmi script, given their historical lineage. Due to the lack of data availability, the study considered a data augmentation technique to increase the number of samples of each class. After that, a new process suggested further increasing the accuracy of the Brahmi text recognition system from the existing studies. At the end of the chapter , automatic sample storing, labelling and the establishment of a statistical distribution are done.

CHAPTER 5 discusses the outcomes of the introduced segmentation methodology, which are meticulously examined across both the Brahmi and Tamil datasets. Additionally, an in-depth evaluation is conducted to assess the performance of the innovative two-phase approach. To ensure the robustness of the proposed recognition system, testing is meticulously carried out on a distinct and smaller subset of the dataset. This specific subset

is formed using five distinct data augmentation strategies, underscoring the commitment to consistent system performance.

The chapter culminates by subjecting the entire system to a comprehensive evaluation, wherein the input is provided in textual form, and the response of system involves accurately allocating characters to their respective folders. This overarching evaluation highlights the proficiency of system in handling real-world input scenarios while maintaining the integrity of character organization and storage..

CHAPTER 6 concludes the research by recapitulating the objectives of Chapter 1 and discussing how each was achieved. This chapter also outlines the contribution and limitations of the study and suggests other aspects of the subject domain that would require future research.

CHAPTER 2

WRITING SYSTEMS AND SCRIPTS OF THE WORLD

This chapter examines the writing systems of the world and scripts, focusing on the history of the Brahmi script, characteristics and properties. The features of all the characters of the Brahmi words are then explored.

2.1 The Writing System of the World

In character recognition, the main distinguishing features are the characteristics of the different writing systems and the structural properties of the characters. According to Ghosh et al. (2010), there are six prominent writing systems; these are listed in Figure 2.1. Further details of these six writing systems are explained below.

2.1.1 Logographic System

A writing system that is based on a logogram or an ideogram is called a logographic system. Ideograms or logograms use graphical symbols to represent a whole word. Examples of writing systems that use logograms include Egyptian hieroglyphs and Chinese characters.

2.1.2 Syllabic System

The logographic system uses a single symbol to represent the entire word. The syllabic system, on the other hand, uses a set of written symbols to represent or approximate syllables or phonic sounds, to make up words. An example of a syllabic system is the Japanese syllable, which includes the Katakana and Hiragana, which use Kana symbols.

2.1.3 Abjad System

The writing system can include the Logographic system such as Chinese (images that represent words or ideas, sometimes with a phonetic element denoting the pronunciation); alphabetic systems (consonant alphabets such as Arabic, or phonemic alphabets with vowels,

such as Latin); and syllabaries (with symbols for consonants and vowels together), such as Ethiopic words (Bosworth, 2003). This means that an alphabet is a set of characters representing phonemes of a spoken language. The term "alphabetic" is used over "phonemic" or "segmental" because the latter terms carry irrelevant theoretical baggage. The word "phoneme" is commonly used in connection with the fact that related but distinct sounds may 'count as the same' with a particular language (Allan, 2015).

The Alphabetic system is sub-divided into Abjad, Abugida, and True Alphabetic systems. The Abjad system merely has symbols for consonantal sounds. This writing is identical to the alphabetic system. The Abjads write from right to left, which is a distinguishing feature of Abjad-based scripts in pen computing (Ghosh et al., 2010). There are two significant scripts in the Abjad system, i.e., Arabic and Hebrew. Typically, an Arabic character involves a long stroke with one to three dots. The written Arabic text appears to be cursive with connecting characters. This unique appearance forms the basis of Arabic script recognition. Urdu, Sindhi, Jawi, and Farsi (Persian) originate from the Arabic script. Conversely, Hebrew character strokes are comparatively consistent in length, and the letters are not connected.

2.1.4 Abugida System

Nearly all the scripts in Southeast Asia and India belong to the Abugida system. The alphabetic-like writing system in the Brahmic family of scripts was originally acquired from the ancient Indian Brahmi script. Figure 2.1 shows the evolution of the original Brahmic scripts, which were popular in India and Southeast Asia in the past.

The northern group of Indian scripts is identical to the original Brahmi script. These scripts include Oriya, Bengali, Gujarati, Manipuri, Gurumukhi, and Devanagari. In contrast, Southern Indian scripts, namely Malayalam, Telugu, Kannada, and Tamil, as well as those

in Southeast Asia, namely Balinese, Lao, Thai, Javanese, and Burmese, have gone through many transformations, and thus are not identical to the Northern India scripts. A special feature of Bengali, Gurumukhi, Devanagari, and Manipuri is that every single character is written without spaces, e.g., the top bar of a word is closely linked. These features create a headline for each word termed as the “Matra” or “Shirorekha”. Hence, these scripts are distinctively identifiable from the appearance of numerous horizontal lines in the text document.

2.1.5 True Alphabetic

This system refers mainly to the Roman script, with English as the most popular script or language worldwide. The original alphabetic system contains letters split into consonants and vowels.

2.1.6 Featural System

The featural system includes phonemes composed of symbols or characters, which represent the features. Korean Hangul is prominently based on the featural system. Mixing logographic Hanja with featural Hangul, the Korean script consists of more Hangul than Hanja, as depicted in Figure 2.1. Korean script is comparatively less complicated and less compact than the Japanese and Chinese scripts, which have more ellipses and circles.

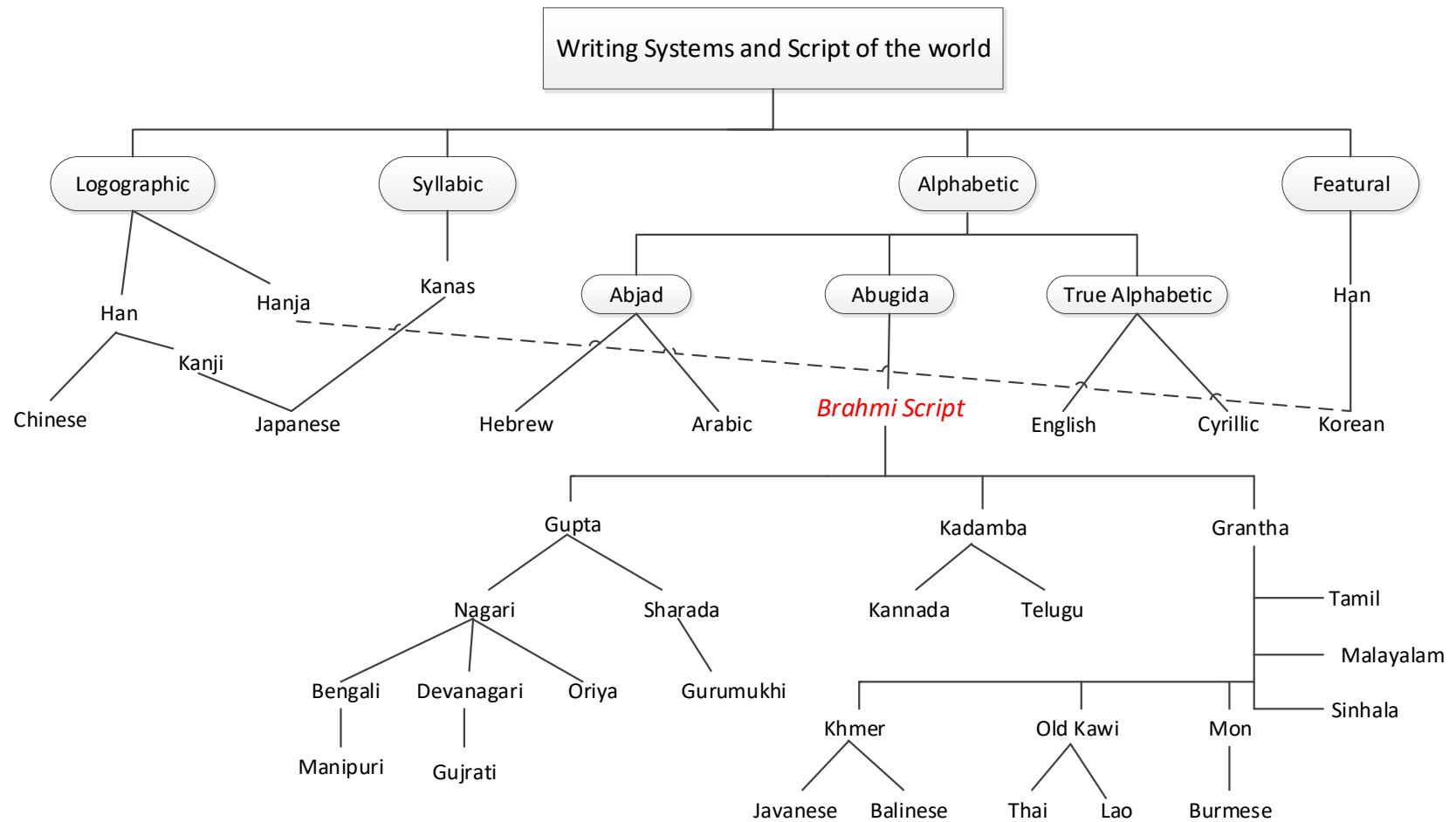


Figure 2.1: The Writing Systems of World (Ghosh et al., 2010; Obaidullah et al., 2018)

2.2 Overview of the Brahmi Script

Brahmi script is an ancient script. Brahmi script was used from the 3rd BCE (Before the Common Era) to the 6th century (CE) (Singh, 2008; Seland, 2014). This script was used to write various religious books on Buddhism and Jainism culture (Salomon, 1998; Subba, 2008). At one time, the Brahmi script was referred to as the "pin-man" script, that is, the "Stick Figure" script in English (Salomon, 1998; Keay, 2008). In other words, Brahmi characters have geometric features such as lines, curves, corners, and dots (Gautam & Chai, 2017) and these features can help identify the Brahmi script from other various scripts (Figure 2.2).



Figure 2.2: Characters and Vowels of the Brahmi Script (Roy & Mandal, 2016)

2.2.1 Properties of Brahmi Script

- i. When a consonant and a vowel come together sequence-wise, one or more extra features may be added to that consonant, and new features can be added anywhere. This extra feature is called "matra" and can be seen inside or outside the characters. If one vowel is followed by one consonant and more than one vowel is followed by one consonant, the new character will be called a compound character and complex

compound character, respectively (Pal & Chaudhuri, 2004). For example, all vowels in Figure 2.3 follow a consonant  and created compound characters that are presented in Figure 2.4.

- ii. The total number of consonants, vowels, and compound characters in Brahmi script is 33, 10, and 325, respectively. That means the total number of characters in the Brahmi script is 368 (Siromoney et al., 1983; Roy & Mandal, 2016).
- iii. The writing style of the Brahmi script is left to right.

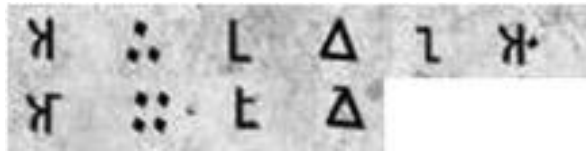


Figure 2.3: Vowels of the Brahmi Script (Roy & Mandal, 2016)

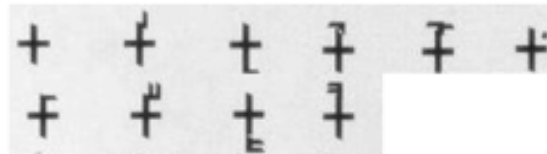


Figure 2.4: Example of Consonant and Compound Characters (Roy & Mandal, 2016)

Figure 2.5 and Figure 2.6 show an example of Brahmi text where different types of characters, compound characters, and complex compound characters are presented.

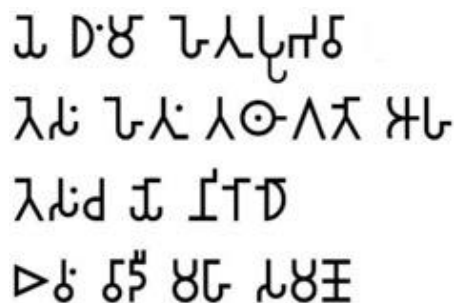


Figure 2.5: Example Text in Brahmi Script (Rajan, 2010)

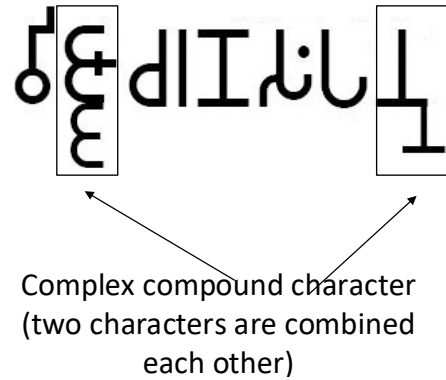


Figure 2.6: Example of Complex Compound Characters

2.2.2 Feature of Brahmi Words

Each vowel shows a unique feature during the creation of compound characters and that feature is called “Matra”. Some examples of the creation of compound characters using vowels and consonants are listed in Table 2.1, where the box (□) shows the “Matra” features. “Matra” is an important feature of the compound character, and the “Matra” feature can help differentiate between a consonant and a compound character, and between one compound character with another compound character. The position of “Matra” in compound characters and its structure when all 10 vowels follow a consonant are explained below from points 1 to 10 (Roy & Mandal, 2016). As each vowel represents a different “Matra”, the 10 rules are explained according to the 10 vowels. All rules and Table 2.1 are inspired by the “Brahmi - Rediscovering the Lost Script” book by Roy & Mandal, (2016).

1. As can be seen, if the  is combined with a , it will not take a modified shape like , meaning that when a consonant is merged with the  vowel, the shape of the consonant will not change.
2. If the  is combined with a , the shape changes into , meaning that when a consonant is merged with the  vowel, the shape change will occur at the top right corner of the consonant.

3. If the  is combined with a  vowel, the shape is changed into , meaning that when a consonant is merged by a  vowel, the shape change will occur at the top right corner of the consonant.
4. If the  is combined with a  vowel, the shape will change into , meaning that when consonants are followed by the  vowel, the shape change will occur at the top right corner of the consonant.
5. If the  consonant is followed by an  vowel, the shape will change into , meaning that when consonants are followed by the  vowel, the shape change will occur at the bottom right of the consonant.
6. If the  consonant is followed by a  vowel, the shape will change into , meaning that when consonants are followed by the  vowel, the shape change occurs at the bottom right of the consonant, and is a little bit similar to that of point number 5. The last compound character has one extra line at the bottom right, and in this case (point number 6), two lines were added to the bottom right side.
7. If the  consonant is followed by a  vowel, the shape will change into , meaning that when consonants are followed by the  vowel, the shape change mainly occurs at the top left side and with one extra line added.
8. If the  consonant is followed by a  vowel, the shape will change into , meaning that when the consonants are followed by the  vowel, the shape change occurs at the top left side and is a little bit similar to that of point number 7. The last compound character has one extra line at its top left side, and in this case (point number 8), two lines are added to the top left side.

9. If the  consonant is followed by a  vowel, the shape will change into , meaning that, when consonants are followed by the  vowel, the shape change is seen at the top left side and the top right side, indicating that this feature is a combination of the “Matra” from point numbers 2 and 7.
10. If the  consonant is followed by a  vowel, the shape will not change, but a new element will be added to the character to turn it into , meaning that when consonants are followed by the  vowel, the shape will not change, but a new isolated element will be added.

Most of the compound characters in the Brahmi script adhere to these ten conditions, but not always. The position of the “Matra” will always depend on the vowel but will also depend on the shape of the consonant. For example, the  consonant, if followed by the  vowel, will result in the “Matra” appearing at the middle right side of , making . However, this “Matra” should appear at the top right side of the consonant, according to Table 2.1. Thus, the position of the “Matra” also depends on the shape of the consonant.

Table 2.1: Example of Compound Characters

Consonants	Vowels	Compound characters	Features
			
			
			
			
			
			
			
			
			
			

2.3 Summary

The writing system of the world is divided into six categories (Logographic system, Syllabic system, Abjad system, Abugida system, True alphabetic system, and Featural system) and each category has its own writing style, property, and characteristics. The Brahmi script is part of the Abugida system and originates from the Brahmic family; it is also the basis of 198 scripts such as Devanagari script, Bangla script, Tamil script, etc. indicating that the Brahmi script has a very rich history. Therefore, properties (details of the characters, creation of compound characters, and the writing method of Brahmi script), characteristics such as “Matra”, placement of the “Matra”, and the visible effect of the vowel in compound characters are important parts of the Brahmi script.

CHAPTER 3

LITERATURE REVIEW

This chapter is dedicated to the assessment of creating the system to build a Brahmi dataset where labeling is done automatically. In order to identify the datasets pertinent to this study, an investigation was conducted into a diverse range of completed research work related to various forms of recognition systems. Additionally, a comprehensive examination was undertaken of multiple studies concerning the segmentation strategies applied to both the Brahmi script and analogous scripts. The primary objective was to identify an optimal approach for effectively segmenting all characters within Brahmi text while rectifying limitations inherent in prior studies. Moreover, the study employs data augmentation techniques as a means to elevate the quality of the samples. Lastly, a critical evaluation was carried out on published studies involving similar recognition systems. The aim was to gauge the accuracy of current word recognition systems when applied to the Brahmi script and to enhance the overall performance of the prevailing Brahmi text recognition system.

3.1 Brahmi Characters and Words Recognition System

Only a few prior studies that developed Brahmi script recognition systems concentrated on Brahmi character recognition, with even fewer claiming to have worked on Brahmi word identification. Gautam et al. (2016) focused on Brahmi character recognition. The zone method was used to extract the features of the Brahmi characters, and the template matching method (coefficient correlation) was used for the classification of all extracted features, achieving an accuracy of 88.83%. However, this method could not recognise non-connected characters. Another study created a system to recognise Brahmi characters (Gautam & Chai, 2017), focusing on the geometric elements in each zone (corner point,

bifurcation point, ending point, intersect point, circle, and semi-circle) and the geometric features (corner point, bifurcation point, ending point, intersect point, circle, and semi-circle) to extract the features. In addition, several sets of criteria for classifying all of these characteristics were developed, resulting in a final accuracy of 94.10%.

Apart from Brahmi character recognition, Brahmi script recognition has a long history, dating back to 1983. Siromoney et al. (1983) in this year developed a system to recognise all Brahmi characters, including all 368 standard vowels, consonants, and compound characters. Their work, however, overlooked all complex compound characters. A coded run method was used for the recognition of machine-printed characters of the Brahmi script, with each Brahmi character manually changed into a rectangular binary array. However, because the suggested technique was generic, it may be used to recognise different scripts. Therefore, not all Brahmi script structural features were the study focus; rather, the study focus was on features that are a part of Brahmi script and other scripts.

Soumya and Kumar (2015) concentrated on recognised the ancient script in the time of King Ashoka (Brahmi script) using Gabor features and zone-based feature extraction methods to extract the features of these characters and next used Artificial Neural Network (ANN) for features classification. Gabor features and zone-based feature extraction methods were used to recognize the ancient script in the time of King Ashoka (Brahmi script), ANN were used to classify the extracted features and the accuracy achieved by this study was 80.2% (Soumya & Kumar, 2015). Next, Vellingiriraj et al. (2016) worked to develop a system to recognise the Brahmi words where zonal density and ANN were used for feature extraction and classification, respectively. To assess the effectiveness the system, three types of data were used: vowels, consonants, and consonantal vowels (compound characters), with accuracies of 93.45%, 93.45%, and 90.24%, respectively. The authors mentioned that

Brahmi text recognition is a tough task as the compound characters, which are consonantal vowels, have similar structures. Thus, there is still a need for research into improving Brahmi text recognition.

As all the studies on Brahmi script recognition employed local datasets to evaluate system performance, the systems did not provide a transparent platform for comparing results. Additionally, Vellingiriraj et al. (2016), who worked on Brahmi text recognition, stated that an official dataset for the Brahmi script was necessary to evaluate the effectiveness of Brahmi script recognition algorithms. Thus, the literature on datasets from various recognition systems was further investigated in order to establish the best approach for creating the Brahmi dataset in this work. The process of constructing the dataset involves a sequence of actions: data collection, pre-processing, segmentation, data augmentation, recognition, storage, and labelling. Except for the initial data collection phase, each subsequent step is designed to be automated. Given the limited familiarity with the Brahmi script among researchers, this automation proves invaluable for future scholars, facilitating their endeavours in this specific domain of research.

Segmentation and recognition assume a pivotal role in this context, given its profound significance in isolating characters within the Brahmi text and recognising them. The quality of both tasks directly impacts the overall performance of the system. Thus, the performances of the existing segmentation and recognition system of Brahmi script will be analyzed and form the basis of the proposed methodology in this study.

3.2 Dataset

Many scripts have already been recognised by automatic recognition systems, with some works also developing systems to recognise ancient scripts. A dataset is essential in the development of a recognition system. This literature study examines the existing datasets

linked to several common scripts of the Abugida group of northern Indian scripts that may be utilised for offline recognition.

The Abugidas include the extensive Brahmic family of scripts from Northern Indian script. It includes several scripts, including the Brahmi script, as well as Devanagari, Bangla, Gurmukhi, Gujarati, and others. Numerous works related to dataset of this system are discussed. More information about these datasets can be found below. A large collection of Bangla numerals (Chaudhuri, 2006) was created, which included roughly 23,392 offline isolated numerals and 8,348 online numeral strings. This Bangla numeral dataset was created using a huge number of postal papers (Chaudhuri, 2006).

One recent study on the Bangla character dataset (Obaidullah et al., 2018) collected 59,892 characters as samples, with 41,536 compound characters, 15,000 primary characters, as well as 3,360 vowels and consonants. Another research (Das et al., 2014) presented a benchmark dataset of isolated Bangla handwritten compound characters. The dataset consisted of a total number of 55,278 separated characters, with 199 different pattern forms and 171 different character classes. In the year 2012, the CMATERdb1 dataset, i.e. Centre for Microprocessor Application for Training Education and Research (CMATER), db stands for database, and the numeric value 1 represents the handwritten line segmentation database prepared for the current work (Sarkar et al., 2012), was developed by Jadavpur University, Kolkata, India. This dataset used 150 page-level handwritten documents, of which 100 pages were written in Bangla script while the remaining pages were written in Bangla text mixed with English words.

To introduce a character-level Devanagari script dataset (Dongre & Mankar, 2012), 750 writers contributed to writing 5,137 Devanagari numerals and 20,305 Devanagari characters. Another dataset of Devanagari script, CPAR-2012 (Centre for Pattern Analysis

and Recognition-2012) (Kumar et al., 2013), was developed, which included 35,000 isolated handwritten numerals, 83,300 characters, 2,000 constrained handwritten pangrams, and 2,000 unconstrained handwritten pangrams.

All the studies mentioned previously confirm that most research has been conducted on contemporary scripts such as Devanagari, Bangla, Gurumukhi, and so forth. This discussion has led to the conclusion that no study has presented a Brahmi dataset. Only a few research have looked into Brahmi character recognition (Gautam et al., 2016; Gautam & Chai, 2017) and Brahmi text recognition (Siromoney et al., 1983; Soumya & Kumar, 2015; Vellingiriraj et al., 2016) but all of them utilised local datasets to train the system. Nonetheless, Vellingiriraj et al. (Vellingiriraj et al., 2016), who worked on Brahmi text recognition, stated that in order to evaluate the effectiveness of Brahmi script recognition algorithms, an official dataset for Brahmi scripts would be necessary.

The research mentioned above followed a few general phases, beginning with the collecting of usable samples and progressing to the conversion of those samples into datasets in order to develop the recognition system. These stages are as follows: gather samples using various accessible resources, separate the useful entities from the obtained samples, organise all segmented characters in a helpful manner (group), and label each group to aid identify the details of the group (Obaidullah et al., 2016; Obaidullah et al., 2018).

Numerous research studies have concentrated on the implementation of data augmentation techniques, which serve the dual purpose of enhancing both the quality and quantity of samples. These techniques play a pivotal role in validating the efficacy of machine learning models. For instance, Yang (2020) leveraged the GridMask-based data augmentation method to elevate the recognition performance of Bengali handwritten graphemes. Additionally, data augmentation techniques have been instrumental in

broadening the spectrum and increasing the sample size, thus bolstering the efficiency of recognizing handwritten medical words, as observed in the work of (Tabassum et al., 2022).

Majid and Smith (2019) also contributed to this domain by applying three fundamental yet highly effective data augmentation techniques, encompassing scaling along the X-axis, shearing along the X-axis, and rotation, as part of their efforts to recognize Bangla Handwritten Characters. Furthermore, Nasim et al. (2020) conducted a comprehensive study assessing the effectiveness of data augmentation in recognizing Bangla Handwritten Digits. Their investigation revealed that the proposed model achieved an outstanding classification accuracy of 99.46% when applied to augmented datasets, as opposed to a 96.72% accuracy for non-augmented datasets. These findings underscore the pivotal role of data augmentation in enhancing the recognition performance of machine learning models.

Maharana et al. (2022) mentioned that machine learning applications across various technology domains and their practical application in addressing real-world challenges are experiencing a rapid and expansive proliferation. The effectiveness of machine learning models is intricately linked to the volume, quality, and variety of the data they operate on. To bolster the reliability of these algorithms, the process of data selection becomes paramount, wherein specific data is discerningly chosen from the original dataset. It is worth noting that contemporary machine learning models often demand high-quality annotated data to guarantee optimal performance (Mumuni & Mumuni, 2022). Thus, data augmentation should be used to enhance the quality and quantity of the proposed Brahmi dataset.

3.3 Segmentation

Several research studies have explored the field of Brahmi character and word recognition. Many of these studies have focused on training and evaluating systems using isolated characters (Roy & Mandal, 2016; Gautam & Chai, 2017; Singh & Kushwaha, 2019; Nagane & Mali, 2020). However, to conduct a comprehensive analysis of an automatic recognition system for Brahmi words, it is crucial to address the segmentation of Brahmi text.

Only a few studies have specifically concentrated on the segmentation of Brahmi characters and words. For example, Gautam et al. (2016) employed line detection algorithm to segment Brahmi characters. However, this approach was unable to isolate certain characters (e.g., ് and ്) due to the absence of physical connections between all the features of these characters. Consequently, the method could only segment characters (vowels and consonants) whose features were physically connected. Singh and Kushwaha (2019) adopted the Horizontal projection profile technique for line segmentation and the Vertical projection profile method for segmenting words, characters, as well as connected characters in Brahmi text. Nevertheless, this method did not yield satisfactory performance due to variations in the skew angles of different characters. Similarly, this method could only segment characters with physically connected features. The authors acknowledged that there is currently no standardized method for isolating Brahmi characters, making it an ongoing research challenge (Singh & Kushwaha, 2019).

Nagane and Mali (2021) focused exclusively on characters that were physically connected to each other in the Brahmi script. Their study first identified words that were physically connected and then isolated them. By utilizing an ensemble classification technique, this approach achieved 100% accuracy in identifying touching characters and

99.16% accuracy in segmenting those characters. However, it is worth noting that this study did not address the segmentation of non-connected characters, which form a significant portion of the Brahmi script. Furthermore, in the context of printed Brahmi text, the characters do not exhibit physical adjacency.

In another study by Nagane and Mali (2020), the authors worked on segmenting Brahmi characters from text, separating them into line and character segments. This study successfully segmented all lines from the text with 100% accuracy and achieved 88.68% accuracy in character segmentation. However, this entire approach relied on the assumption that the characters would have consistent sizes. Consequently, if the data is collected from various sources and the character sizes vary, it may impact the performance of segmentation approach.

The Brahmi script, similar to the English script, features characters that are not physically connected to each other. Moreover, the Brahmi script includes the distinctive "matra" feature, which is also present in derived scripts like Devanagari, Bangla, Tamil, and Telugu.

Several approaches have been employed for Devanagari text segmentation. Srivastav and Sahu (2016) achieved a 90% accuracy rate by utilizing vertical and horizontal projection profiles of individual characters. Sahare and Dhok (2019) adopted a hybrid approach that combined projection profiles, graph distance theory, and an SVM classifier to segment both Devanagari and Latin scripts. This approach resulted in accuracy rates of 97.32% and 98.79%, respectively. However, their study revealed a relatively high bad segmentation rate of 2.50%, which was significantly higher than the over-segmentation rate of 0.18%.

Narang et al. (2020) conducted a study aimed at segmenting lines from ancient Devanagari texts. Their method involved dividing the document image into vertical stripes

and employing piecewise horizontal projection profiles, achieving an accuracy rate of 97%. However, this study did not utilize a dataset and did not focus on character segmentation within the lines.

In another study by Narang et al. (2019), the authors aimed to segment characters from ancient Devanagari texts. Their approach involved dividing the document image into vertical stripes and using piecewise horizontal projection profiles for line segmentation. The lines were then further segmented into words using vertical projection profiles. Finally, an iterative algorithm called the drop flow method was employed to segment words into characters, resulting in an overall accuracy of 96%.

To summarize, it can be observed that most studies have primarily focused on utilizing the projection profile approach for character segmentation in Devanagari text. Moreover, the Bangla script shares similarities with the Devanagari script. Ahmed et al. (2019) suggested an approach for segmenting Bangla words from text, where the Bangla text can be divided into three zones. The upper zone represents the portion above the headline, the middle zone encompasses the basic characters or compound characters below the headline, and the lower zone is the area where modifiers may reside. In this study, a Convolutional Neural Network was utilized to build a model that identifies individual Bangla characters. The lines and words were segmented from an image based on empty spaces, achieving 100% accuracy in line segmentation and word segmentation. However, the study did not specifically focus on isolating characters from words. Ferdous et al. (2021) focused on the segmentation of Bangla compound characters. In this case, the characters were already written separately and not physically connected, making it easier to identify the segmentation points between them. Segmentation points refer to the points that help isolate two characters.

Abir et al. (2020) employed pixel values and specific rules to perform line segmentation of text and word segmentation within the lines of the Bangla script. For character segmentation from words, the study involved two steps: removing the "matra line," which helps eliminate physical connections between characters, and then isolating each character using vertical lines. The system achieved accuracy rates of 100% for line segmentation, 99.10% for word segmentation, and 94.32% for character segmentation. Furthermore, Pramanik and Bag (2020) proposed a system that first detects and corrects skew in handwritten Bangla and Devanagari words, estimates the headline, and subsequently segments the words into meaningful pseudo-characters. This process is followed by the extraction of three statistical features and their combination with off-the-shelf classifiers to study and identify exemplary combinations. The achieved accuracy rates for segmenting Bangla and Devanagari characters in this study were 94.01% and 93.32%, respectively.

Gayathri Devi et al. (2022) introduced a segmentation scheme called Text Line Slicing (TLS) to segment palm leaf characters. The TLS algorithm proposed in this study utilizes four variables, namely Vertical space (Vs), Horizontal space (Hs), Vertical Track (VT), and Horizontal Track (HT), to identify the extension of character strokes. Similarly, another study also explored changes in the TLS approach to extract all Tamil characters from the text (Sathik & Ratheash, 2021). However, both approaches based on TLS have a limitation in that they can only segment characters whose features are physically connected. In other words, if a character does not have all its features physically connected, the TLS approach would consider it as two separate characters. It should be noted that, in the Brahmi script, there are certain characters where the features of a character are not physically connected.

Maheswari et al. (2022) employed a segmentation process consisting of multiple sub-processes, including profile smoothing, over-segment reduction, segmentation of touching and broken characters, and character segmentation. This approach was utilized to accurately extract individual characters from stone inscriptions.

In addition to these techniques, machine learning, deep learning, and object detection methods have also been utilized to identify the character areas within the text and extract the characters based on regions of interest (ROI) such as Preethi and Mamatha (2023) directed their efforts towards segmenting text in digitized inscriptional images. This study employed a region-based convolutional neural network (CNN) to detect characters within the images. The segmentation results achieved by InceptionV3 were recorded as 74.79% for printed de-noised epigraphical images, 71.53% for de-noised epigraphical images, and 18.11% for noisy images. Similarly, Abbass and Marhoon (2022) focused on extracting characters from Iraqi car license plates using the find contours approach, which is based on object detection. Likewise, Siddharth et al. [36] aimed to extract URLs from images by employing line, word, and character detection techniques based on horizontal projections, vertical projections, and finding contours, respectively. Since URLs consist of English characters and numbers, which are not physically connected to each other.

Furthermore, Rajyagor and Rakholia (2021) concentrated on segmenting characters from Gujarati text. Researchers employed a hybrid method for tri-level segmentation, involving line, word, and character segmentation from the image. The horizontal projection technique was used for line segmentation, the scale-space approach including finding contours and connected components was employed for word segmentation, and the vertical projection technique was utilized for character segmentation.

Ayyoob and Ilyas (2022) focused on character segmentation in the Vattezhuthu text. Their study also utilized the Find contours technique to identify the region of interest (ROI) for words in the text and extract them. Similarly, Bhatta et al. (2021) employed Find contours to extract characters from license plates. These studies utilized Find contours as an object detection technique capable of detecting individual characters. However, it is important to note that the Find contours works effectively when the objects or characters are not physically connected. Therefore, this approach is suitable for segmenting words or individual characters that are not linked to other characters.

Since all characters and compound characters in the Brahmi script are not physically connected, it can be concluded that the Find contours technique is a suitable option for extracting these characters from the text.

3.4 Approaches for Brahmi Text Recognition: An Overview

This section is split into two segments. The initial part explores the suitability of the holistic and analytic approaches for Brahmi text, while the second part concentrates on the single-phase and two-phase methods, in conjunction with a classifier, for Brahmi text analysis.

3.4.1 Holistic and Analytic Approaches

To recognise the Bangla and Devanagari words, a holistic analysis technique was utilised. Bhowmik et al. (2012) used this approach to recognise the 119 types of words of the Devanagari script and achieved 96.20% recognition. However, the types of meaningful words in the Devanagari script are countless. As a result, this study relied on a small number of samples to assess the performance of the suggested system. Similarly, Bhowmik et al. (2014), Adak et al. (2016), Bhowmik et al. (2019) also focused on recognising the Bangla words by using a holistic approach and achieved 85.88%, 85.42%, and 79.38%, respectively.

In addition to Bangla word recognition, a holistic approach was employed for recognizing Devanagari words, achieving an accuracy rate of 86.14% (Roy et al., 2016). To increase the performance of the holistic approach, the word prediction approach is also used for the whole word recognition. In this approach, words are predicted based on the combination of characters. A combination of the holistic approach and word prediction approach are used to recognise the Bangla and Devanagari words. Sánchez and Pal (2016) used the Hidden Markov Model (HMM) and *n*-gram language model and achieved 80.60% accuracy. Similarly, Ramachandrule et al. (2012) recognised the Devanagari words with 94.23% accuracy using the two-phase Dynamic Programming algorithm to match the test word. However, the holistic approach merely addresses the issue of a fixed or limited vocabulary. For big vocabulary issues, this method may be utilised to decrease the lexicon (Madhvanath et al., 1999; Madhvanath & Govindaraju, 2001). The holistic approach also necessitates the use of a dictionary or book, which can assist in providing information about all potential words or vocabularies. However, no official dictionary for Brahmi words or such book on the meaning of Brahmi script could be identified during the review undertaken in this study. Because accessible examples of Brahmi words and text are scarce, the holistic method may be less suited for a Brahmi word identification system than the analytical technique.

As previously mentioned, the analytical approach was found more suitable than the holistic approach for the Brahmi word recognition system in this study. In the analytical approach, to separate the characters from the words in the analytical method, two ways can be used: 1) All compound characters are isolated from the words, and each compound character is recognised independently; 2) All "matra" are isolated from the words, and all isolated "matra" and characters are recognised separately. Based on these two approaches, all words were first split into vowels, consonants, and "matra" and then used in the

recognition system. Table 3.1 presents the comparison of all types of approaches (Holistic and both types of Analytical approaches) in terms of Brahmi script.

Table 3.1: Comparison of the Three Approaches in Terms of Brahmi Script

Advantage	Limitation	Comment
Holistic approach		
1. Focused on recognising the whole word. 2. Useful for recognising whole words. 3. Whole word prediction aids performance increase.	1. Requires knowledge of all possible words. 2. Dictionary is required to predict words. 3. Numerous classes involved.	There is no official Brahmi dictionary and no compilation of information on the meaning of Brahmi words.
Analytical approach (all compound characters are isolated from the words)		
The restriction of the compound characters is that “matra” cannot be separated from such characters. Removing the “matra” from a particular compound character would alter the meaning of the character. Therefore, this method may be used to identify compound characters.	If there are numerous classes, this method might affect system performance.	The Brahmi script contains 368 compound characters. Given this large number, this technique may not be appropriate for recognising compound letters in Brahmi words.
Two-phase analytical approach (divided into three zones and all “matra” are isolated from the words)		
1. Converts compound characters into “matra”, vowels, and consonants. Therefore, it reduces the number of classes. 2. Can work in two phases, which aids increased performance and decreases the number of classes.	Not useful if “matra” cannot be isolated from compound characters.	1. “Matra” cannot be separated from text of Brahmi script by a line. 2. The two-phase process can be used to recognise Brahmi compound characters, which are created by consonant and vowel combinations.

Both approaches of analytical methods are used to recognise the northern Indian scripts. The first analytical technique focuses on recognising compound characters in text, and some research has also concentrated on compound character and word identification in northern Indian scripts (Bangla, Devanagari, Gurumukhi and Gujarati script). For example, zone, and row-wise longest-run with Support Vector Machines (Radial Basis Function) (SVM (RBF)) (Sarkhel et al., 2015) were used for the feature extraction of Bangla compound characters, achieving 72.87% accuracy. Das et al. (2010) used shadow features, longest run features, and quad-tree-based features of the Bangla compound characters and achieved 79.25% accuracy using the Multilayer Perceptron (MLP) classifier and 80.51% using the SVM classifier. Structural decomposition was also used to recognise Bangla compound characters, gaining 96.17% accuracy, while template matching was used to classify the features (Bag et al., 2014) by using an official dataset which was created by IIT (Indian institute of technology) Kharagpur, India. Recently, Pramanik and Bag (2018) used a chain code histogram with ANN to recognise the Bengali compound character and achieved 88.74% recognition rate. Similarly, Deep Convolutional Neural Networks (DCNN) was used to train Bangla compound characters and achieved 90.33% accuracy by using CMATERdb 3.1.3.3 database (Roy et al., 2017), which was a good and efficient recognition rate when compared to previous research.

3.4.2 Single-phase and two-phase approach

When compound characters are combined in a meaningful manner, a word is created. Roy et al. (2014) focused on Bangla word recognition via local gradient histogram and sliding window traversal for feature extraction and utilized the Hidden Markov Model (HMM) for classification. The study achieved 67.18% accuracy. Another study used HMM to recognise Bangla words and reached 85.49% accuracy (Roy et al., 2016). Bhowmik et al.

(2014a) applied the elliptical feature extraction method and evaluated the classifiers performance of Naïve Bayes, Bagging, Dagging, SVM, MLP. He reported the accuracies of 74.41%, 60.00%, 69.41%, 77.35%, and 77.94%, respectively. Histogram of Oriented Gradients (HOG) for feature extraction and ANN for the classification of the extracted features were used to recognise Bangla words in one study, which achieved an accuracy of 87.35% (Bhowmik et al., 2014b).

Devanagari compound characters and Devanagari word recognition were also investigated by some researchers. Shaw et al. (2008) used zone, directional chain code, and HMM to classify Devanagari words and achieved 80.2% accuracy. Mahender and Kale (2011) used the combination of some sets of rules to classify the features of Marathi words (Devanagari words), extracted using structural elements, and achieved 80.78% accuracy. Garg et al. (2013) used shape-based features with SVM to recognise the Hindi text and achieved 76.4% accuracy.

According to the preceding explanation, the holistic and analytical approach can have an impact on the performance of a word recognition system. Similarly, both analytical approaches (all compound characters isolated from the words, and all vowels, consonants, and “Matra” recognised separately from the word) can affect the performance of the recognition system. Each strategy has advantages and disadvantages, and each technique performs differently depending on the features and characteristics of the script.

As mentioned in Section 3.1, there are few studies on Brahmi word recognition, such as that of Soumya and Kumar (2015), who concentrated on recognising the Brahmi script using Gabor features and zone-based feature extraction methods to extract the features of these characters and subsequently classified the features with ANN. Their system was 80.2% accurate. Following that study, Vellingiriraj et al. (2016) developed a Brahmi word

recognition system using zonal density to extract the features and ANN to classify the extracted features. The system effectiveness was assessed with three data types: vowels, consonants, and consonantal vowels (compound characters), which yielded 93.45%, 93.45%, and 90.24% accuracy, respectively. Both aforementioned Brahmi word recognition-based studies used an analytical approach where each character was separated from the word and subsequently recognised.

Devi and Sathiyapriya (2021) utilized ResNet 50 for Brahmi character recognition; however, their study lacks performance evaluation. Similarly, Subadivya et al. (2020) employed another deep learning architecture, CNN, for recognizing Brahmi-Tamil words, achieving an accuracy of 94.6%. Brindha and Bhuvaneswari (2021) employed feature extraction methods such as Zernike moment and zoning features, along with the chi-square test and ANN, to classify the extracted features. On the other hand, Preethi and Mamatha (2023) manually collected a dataset comprising 287 characters with 210 samples per class, creating a sufficiently large dataset for training and testing deep convolutional neural networks. This study applied transfer learning to Alexnet, Googlenet, and Inception-V3 networks for Brahmi input, with Inception V3 achieving an impressive accuracy of 97.62%. Comparing traditional machine learning models trained and tested on Hu-Zernike moments, transfer learning demonstrated superior results by capturing extensive features from image samples.

Northern Indian scripts (Devanagari, Bangla, Gurumukhi, and Gujarati script) also feature the property of compound characters, which is similar to the Brahmi script. Two compound characters, which are made up of a consonant and two different vowels, would be fairly similar. “Matra” is an essential characteristic of the compound character, and it may be observed on both the upper and bottom sides of the consonant. The review discovered

that; an analytical approach was introduced to recognise the northern Indian script. This method improved the efficiency of different recognition systems by removing the word "matra" from compound characters and converting it to a consonant, as well as reducing the number of classes (groups). The text is split into three zones in this method, with "matra" belonging to the lower and top zones and characters belonging to the centre zone. All of the "matra" and characters from the three zones are used together to recognise the text in a single phase. To further increase the performance of this approach, this process was divided into two phases. All "matra" of the lower and upper zones, and all characters of the middle zone are recognised in the first phase and second phase, respectively. Roy et al. (2016), for example, used HMM and divided each Devanagari word into three zones in order to classify the Devanagari words by using the two phases approach and achieved 86.14% accuracy. Ajmire et al. (2015), and Golait et al. (2017) used this two stages approach with one classifier to recognise the Devanagari (Marathi) compound characters. The invariant rotational moment and central moment (translation moment) with SVM and Discrete Wavelet transform with KNN techniques were used separately to train the classifier and achieved 93.87%, 94.92% accuracy, respectively. To further increase the performance, Das et al. (2015) suggested a modification to the analytical approach, where they proposed a two-phase approach with multiple classifiers to recognise Bangla compound characters. The researchers significantly improved the recognition rates of handwritten Bangla characters with this two-phase approach as compared to the traditional single-phase approach. Bangla compound characters are made up of two entities (consonant and "matra").

In the context of employing a two-phase approach within a single script, Bangla digits were recognized using this method, resulting in an improved classification rate of 93.5% in the second pass compared to the 90.5% obtained in the first pass (Basu et al., 2004).

Subsequently, the two-phase approach was employed to recognize Bangla characters, achieving an accuracy of 85.40% (Das et al., 2009). More recently, the two-phase approach was applied to recognize Bangla words, attaining an accuracy of 87.26% (Das et al., 2015). These experiments demonstrated a significant enhancement in recognition rates for handwritten Bangla characters compared to the traditional single-phase approach. This compelling evidence highlights the consistent and better performance of the two-phase approach.

The Bangla script is historically derived from the Brahmi script and shares the "Matra" feature, just like in the Brahmi script. As a result, the two-phase approach is expected to be effective in recognizing Brahmi words as well. However, the process involves separating the "matra" first from the words by dividing the word into three zones, and subsequently, the two-phase approach is utilized to recognize all the characters in the middle zone. Nonetheless, this approach is not suitable for the Brahmi script as "matra" cannot be removed from Brahmi compound characters. Given this unique characteristic, the two-phase analytical approach would be appropriate to recognise Brahmi words. Brahmi words can also be recognised using an analytical technique (all compound characters are separated from the words). Therefore, it is believed that combining the two-phase approach with the aforementioned analytical approach can aid the improvement of the performance of existing Brahmi word recognition systems. With this, this study proposes an approach which combines the two-phase technique with the analytical approach to improve the existing Brahmi word recognition system.

Next to that, there remains a lack of exploration of various deep learning architectures for recognizing Brahmi words. As a result, this study proposes the application of diverse deep learning architectures to effectively recognize Brahmi words. Apart from

that, Memon et al. (2020) highlighted a report based on text recognition challenges that winners used different types of deep learning architectures. Thus, various deep learning architectures such as AlexNet, ResNet can be suitable options for classification.

3.5 Summary

This chapter examined endeavours within the realm of automated Brahmi dataset creation. The Brahmi text recognition system comprises four core components: data collection, segmentation, recognition mechanisms, and the automation of labelling and storage. The observation has been made that prevailing Brahmi text recognition systems frequently rely on localized datasets for assessing system accuracy. However, these localized datasets lack the transparency required for comprehensive result comparison. Hence, the present study endeavours to establish a dataset through automated work. This chapter also underscores the significance of utilizing data augmentation as a means to improve both the quality and quantity of Brahmi character samples. This study reviews the segmentation and recognition stages in tandem with dataset creation. This investigation encompasses a comprehensive survey of diverse methodologies to identify the optimal approach to enhance segmentation and recognition proficiency of the Brahmi text.

Regarding the segmentation of Brahmi text, the review highlights the Hybrid of Object Detection and K-Nearest Neighbors (KNN) approach as the most suitable technique for advancing segmentation efficacy while overcoming limitations inherent in existing segmentation methodologies. The study further proposes a two-phase process as a viable strategy for augmenting the performance of prevailing Brahmi text recognition systems.

CHAPTER 4

METHODOLOGY

The main objective of this chapter is to present a system that can convert Brahmi text into the Brahmi character dataset where character represents, character, compound characters and complex compound character, and perform automatic labelling of Brahmi text. The steps to develop the system include the collection of data, pre-processing including noise removal, isolation of useful characters from the words and texts, data augmentation, character recognition, division of the isolated characters into groups, and execution of the labelling process to store the data in a proper manner for further use.

The complete methodology of the presented model is illustrated in Figure 4.1, showcasing two innovative approaches: a Contour-KNN Brahmi Segmentation (CKBS), and a Two-Phase Enhanced Brahmi Recognition (TPEBR) for segmentation and recognition, respectively. Additionally, this study proposes a model for the automatic data labeling of Brahmi text characters.

4.1 Data Collection

The first step in creating the dataset was to find out the appropriate resources for the samples of Brahmi text. Multiple online sources were considered for the collection of Brahmi words and text as samples in image format. However, all those images, that were clearly visible and noise free, were considered for the sample collection. Figure 4.2 shows the visible and noise-free, as well as noise-containing Brahmi word examples. Following this step, the characters were separated from the words and text in the segmentation process. This is an essential task as these isolated characters would be later used to train the system.

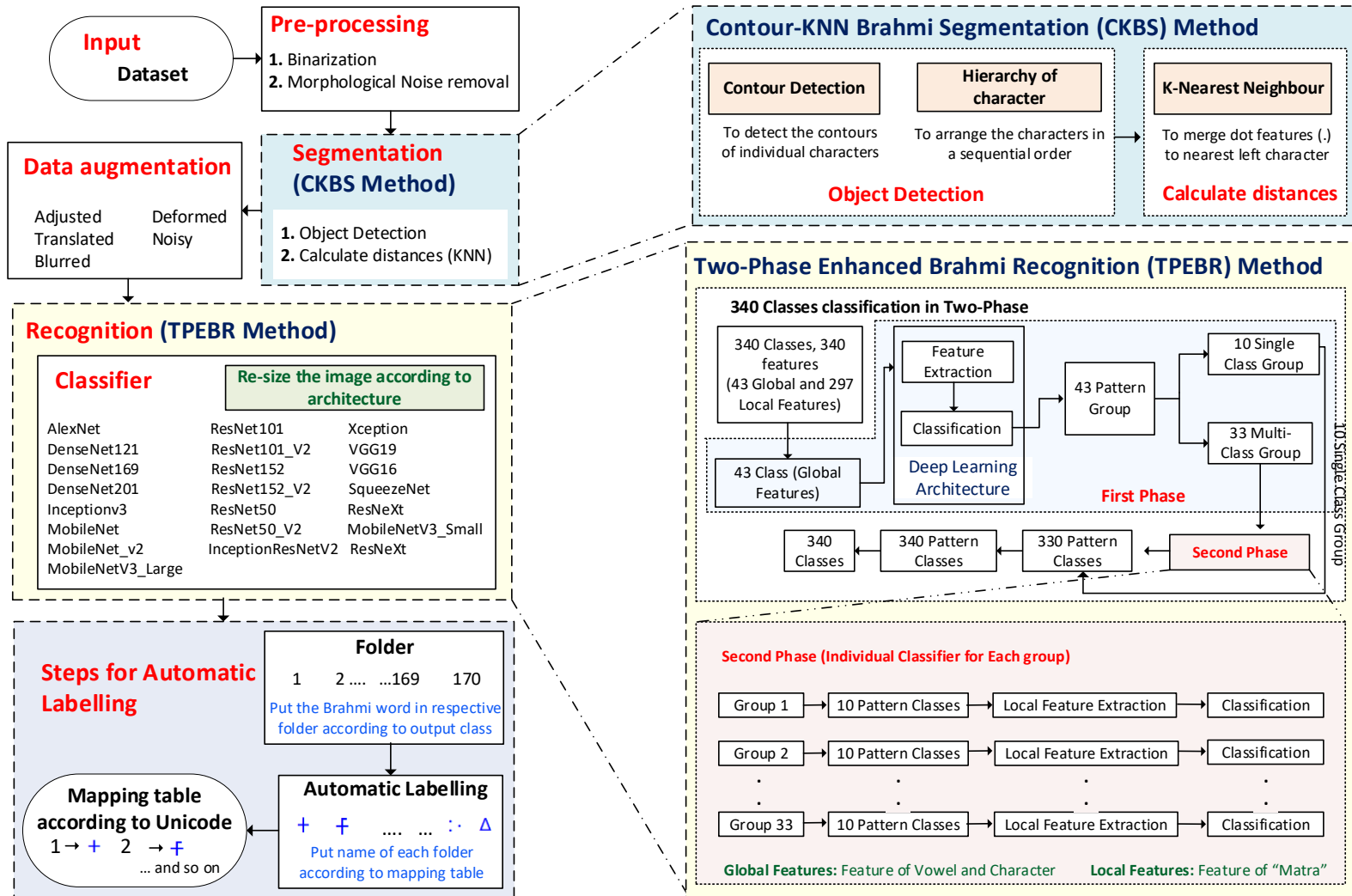


Figure 4.1: A Diagram of the Overall Methodology

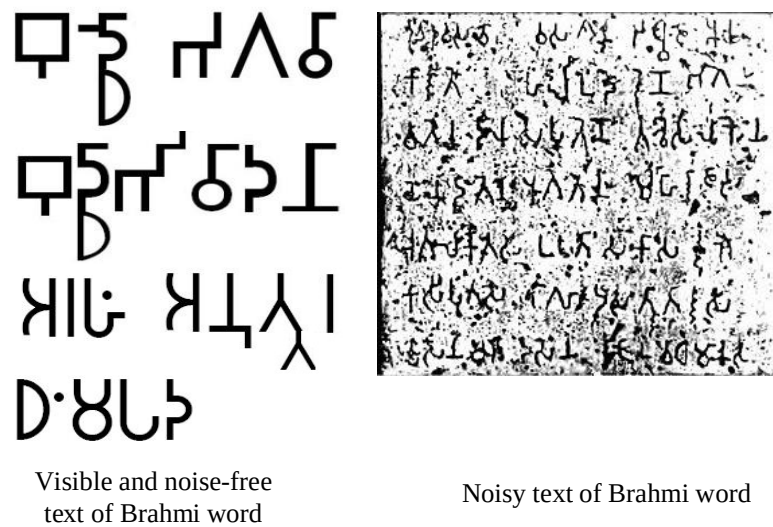


Figure 4.2: Samples of Brahmi Text, With and Without Noise

4.2 Pre-processing

Pre-processing is a very important step that can help enhance input quality. Thresholding and noise removal are the steps which are used by this study, and details of these steps are further elaborated below.

4.2.1 Binarization/Thresholding

Thresholding (binarization) refers to the conversion of a grayscale image into a binary image. The purpose of thresholding is to identify the extent of the object and to concentrate on the shape analysis. In general, thresholding can be done globally or locally.

Global thresholding selects a single threshold value for the whole image based on an assessment of the background level from the intensity histogram of the image. As a result, this approach is beneficial if the background of the image is basic and homogeneous. Local or adaptive thresholding uses different threshold values based on the local area information. This method is typically useful for real-world documents since these documents are sometimes purposefully created with stylish, colourful, and complicated backgrounds. As

the background of all the images in the Brahmi dataset was white, the thresholding process in this study was performed using a global threshold approach which is Otsu's method.

Otsu's method (Yousefi, 2011) divides the image into two zones (T_0 and T_1). The pixel value of T_0 is zero to t and that of T_1 is $t + 1$ to l , where t and l are the threshold value and maximum grey value of the image, respectively. Thus, T_0 and T_1 indicate light and dark regions, respectively, with zero and one represent white and black, respectively. Otsu's method determines the threshold value based on the statistical information of the image where for a chosen threshold value t the variance of clusters T_0 and T_1 can be computed. The optimal threshold value was calculated by minimising the sum of the weighted group variances, where the weights are the probability of the respective groups as follows (Equation 4.1):

$$P(i) = \frac{\text{number}\{(r,c)|\text{image}(r,c)=i\}}{(R,C)} \quad \text{Equation 4.1}$$

Where $P(i)$ is the histogram probabilities of the observed gray value for $i=1,...,L$, r and c are the index for the image row and column, respectively, while R and C are the image row and column number, respectively.

The σ_w^2 is the intra-class variance, where the threshold value for which this variance is lower will be considered the final threshold. $w_b(t)$ (Equation 4.2), $\mu_b(t)$ (Equation 4.4), and $\sigma_b^2(t)$ (Equation 4.6) are the weight, mean, and variance of T_0 , respectively, with intensity values from zero to one, and $w_f(t)$ (Equation 4.3), $\mu_f(t)$ (Equation 4.5), and $\sigma_f^2(t)$ (Equation 4.7) are the weight, mean, and variance of T_1 , respectively, with intensity values from $t + 1$ to 1.

$$w_b(t) = \sum_{i=1}^t P(i) \quad \text{Equation 4.2}$$

$$w_f(t) = \sum_{i=t+1}^l P(i) \quad \text{Equation 4.3}$$

$$\mu_b(t) = \frac{\sum_{i=1}^t i * P(i)}{w_b(t)} \quad \text{Equation 4.4}$$

$$\mu_f(t) = \frac{\sum_{i=t+1}^l i * P(i)}{w_f(t)} \quad \text{Equation 4.5}$$

$$\sigma_b^2(t) = \frac{\sum_{i=1}^t (i - \mu_b(t))^2 * P(i)}{w_b(t)} \quad \text{Equation 4.6}$$

$$\sigma_f^2(t) = \frac{\sum_{i=t+1}^l (i - \mu_f(t))^2 * P(i)}{w_f(t)} \quad \text{Equation 4.7}$$

The Otsu thresholding techniques is based on inter-class variance from the foreground and background regions. This thresholding approach calculates the minimum intra-class variance. The inter-class variance was defined using the following equation (Equation 4.8):

$$\sigma_w^2 = w_b(t) * \sigma_b^2(t) * w_f(t) * \sigma_f^2(t) \quad \text{Equation 4.8}$$

σ_w^2 is the with-in-class variance and for which threshold value this variance is lower, that will be considered as the final threshold.

Following the application of Otsu's thresholding to all the samples, various threshold values are employed to transform the samples into black and white images. Specifics regarding these threshold values can be found in Table 4.1.

Table 4.1: Summary of the Threshold Value is Determined Through Otsu's Thresholding Method for all of the Images

Type	Value
Average Threshold Value	106.66908803789973
Minimum Threshold Value	73.0
Maximum Threshold Value	169.0
Median Threshold Value	127.0
Mode Threshold Value	128.0
Range	[73,169]

4.2.2 Noise Removal

Morphological opening is a commonly used operation in image processing to remove noise and small unwanted objects from binary or grayscale images. The morphological opening involves two fundamental operations (structuring element): erosion followed by dilation.

Morphological opening has been widely employed in various studies, including ID card detection systems (Satyawar et al., 2019) and vehicle number plate recognition (Kumar et al., 2021). In these applications, the Erode and Dilate structuring elements have gained popularity for noise reduction. Similarly, Erode and Dilate operations were applied to eliminate noise from digital Telugu text, which has its origins in the Brahmi script (Edem, 2021; Nagendra Kumar et al., 2022). Given this context, the proposed study also utilized morphological opening, incorporating Erode and subsequent Dilate structuring elements, and utilizing the same 3×3 kernel size as employed by Edem (2021).

Input image, the output image of Erosion, and Dilation as structuring elements are presented in Figures 4.3, 4.4, and 4.5, respectively.

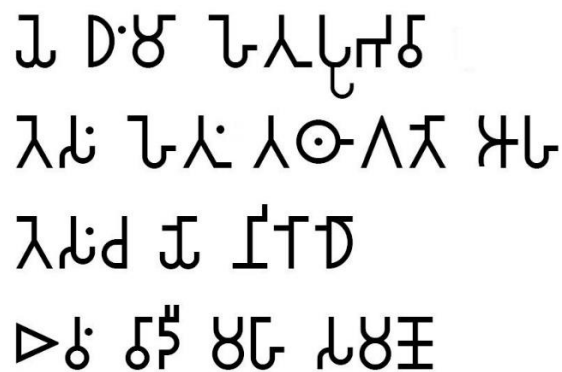


Figure 4.3: Input image

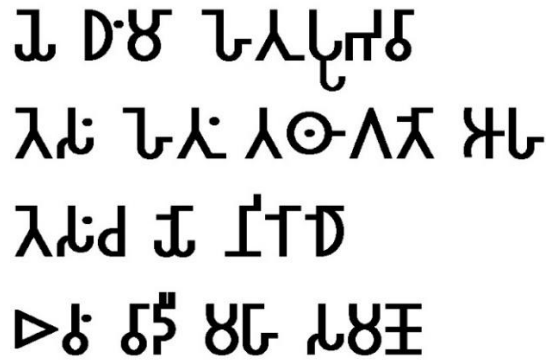


Figure 4.4: Output image of Erosion structuring element

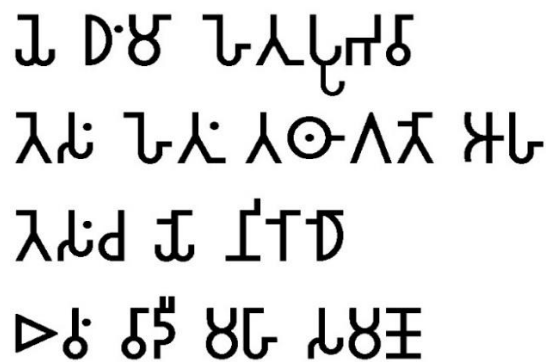


Figure 4.5: Output image of Dilation structuring element

Pseudocode of Morphological Opening is mentioned below.

Algorithm: Pseudocode for Morphological Opening for Noise removal

Input:

- Image: Binary input image
- Kernel: Structuring element (Erode, Dilate) for morphological operations

where kernel size 3×3

Output:

- Result: Binary image after morphological opening.
-

Process:

Repeat:

- # Erode the image using the structuring element
ErodedImage = Erode(Image, StructuringElement)
- # Dilate the eroded image using the structuring element
OpeningImage = Dilate(ErodedImage, StructuringElement)
- # Set the output image to the opening image
OutputImage = OpeningImage

Until no further change is detected in the output image

Return OutputImage

4.3 Contour-KNN Brahmi Segmentation (CKBS) Method

Segmentation is an important stage in the construction of a dataset and a recognition system. The techniques selected for the recognition system depend on the types of the dataset used, such as the isolated characters, full words, or scene samples. Therefore, the type of dataset (isolated characters or full words) and how it influences the Brahmi text recognition system is thoroughly examined.

There are two ways of segmentation: (1) the analytical method (Koerich et al., 2003; Plötz & Fink, 2009) and (2) the holistic approach (Madhvanath & Govindaraju, 2001). The details and requirements of both processes were presented in Chapter 3 Section 3. This study revealed that, the analytical approach (all characters are isolated from the words) was better suited for building the proposed Brahmi dataset in this study.

In this study, a hybrid approach combining object detection and k-nearest neighbors (KNN), termed as Contour-KNN Brahmi Segmentation (CKBS) method, was proposed for character segmentation in Brahmi text.

4.3.1 Object Detection

Object detection was employed as the initial step to identify all the characters within the text. The study utilized the Contour Detection approach to detect the contours of individual characters, and then the Hierarchy of characters was utilized to arrange the characters in a sequential order.

4.3.1.1 Contour Detection

Contour detection with an 8-connectivity neighbourhood is a technique used in image processing and computer vision to identify and extract the boundaries of objects or regions of interest within an image. It provides more detailed information about the object

boundaries by considering diagonal connections in addition to the horizontal and vertical connections.

The explanation of the contour detection process with an 8-connectivity neighbourhood for segmentation of character from Brahmi text:

The initial step involves converting the image into a binary format. Since this conversion has already been accomplished during the pre-processing phase, there is no necessity to repeat this procedure at this juncture. This process distinguishes the character of interest from the background by assigning binary values (typically 0 for background and 1 for foreground) to each pixel based on its intensity or color. Further steps of this approach are explained below:

1. Identify foreground pixels: Once the image is converted to binary, identify the foreground pixels representing the character or regions of interest. These are the pixels with a value of 1 in the binary image.
2. Initialize the contour set: Create an empty contour set to store the detected contours.
3. Traverse the image: Iterate through each foreground pixel in the binary image.
4. Check 8-connectivity neighbours: For each foreground pixel, consider its 8-connectivity neighbours, which include the eight pixels surrounding the current pixel, including diagonal connections. These neighbours are indexed as follows:

1	2	3
4	P	5
6	7	8

5. Identify contour points: Analyse the neighbourhood of each foreground pixel to identify contour points. A contour point is defined as a foreground pixel that has at least one background (0) neighbour. This indicates a transition from the object to the background.

6. Store contour points: Add each identified contour point to the contour set. This set of contour points represents the boundary or outline of the object or region of interest.
7. Repeat for all foreground Pixels: Repeat steps 5-7 for all foreground pixels in the binary image to ensure that all contour points are detected.

By following this process, contour detection with an 8-connectivity neighbourhood allows for more precise segmentation and boundary extraction, considering both diagonal and orthogonal connections between pixels. It captures more detailed shape information and is used to segment the characters according to the outer structure. The algorithm of counter detection is presented further.

Algorithm: Contour Detection with 8-connectivity Neighbourhood for Segmentation

Input: A binary image with height (h) and width (w).

Output: A list of contour sets.

1. Initialize an empty list of contour sets.
 2. Initialize a 2D array or matrix "visited" of size (h, w) to keep track of visited pixels. Set all elements to false initially.
 3. Define the 8-connectivity neighborhood offsets as follows: offsets = [(-1, -1), (-1, 0), (-1, 1), (0, -1), (0, 1), (1, -1), (1, 0), (1, 1)]
 4. Iterate over each row j from 0 to h-1:
 - Iterate over each column i from 0 to w-1:
 - If pixel(i, j) is equal to 1 and visited[j][i] is false, then:
 - Create a new contour set, initialize it as an empty list, and append it to the list of contour sets.
 - Call the function "traceContour(i, j)" with the current pixel coordinates.
 5. Define the function "traceContour(x, y)" to recursively trace the contour:
 - Set visited[y][x] as true.
 - Add the current pixel (x, y) to the last contour set in the list of contour sets.
 - Iterate over each offset in offsets:
 - Calculate the new coordinates $nx = x + \text{offset}[0]$ and $ny = y + \text{offset}[1]$.
 - If the new coordinates (nx, ny) are within the image boundaries and visited[ny][nx] is false, then:
 - If pixel(nx, ny) is equal to 1, call the function "traceContour(nx, ny)" recursively.
 - Else, add the current pixel (x, y) to the last contour set in the list of contour sets as a boundary pixel.
-

As depicted in Figure 4.7, a total of 40 characters were recognized in the object detection phase, treating each dot (.) feature as an independent character. However, a dot (.) does not represent an individual character. This discrepancy arises because the object detection approach treats dot features as individual characters since they are not physically connected to the respective characters. Consequently, it can be concluded that the previous step failed to accurately detect all characters that possess dot features.

In this research work, a few steps were proposed to be taken to address the issue of the dot (.) feature not being considered as part of the previous character in Brahmi text. Firstly, it was observed that the size of the dot (.) character is smaller than the regular characters, and it was determined that the size of the dot (.) should be less than 1/3 of the average size of a character. The size of a character was calculated based on the width and height of the region of interest (ROI), as determined in the object detection stage. The area of a character was calculated using the Equation 4.9.

$$\text{Area of a Character} = \text{Height} \times \text{Width} \quad \text{Equation 4.9}$$

By calculating the areas of all characters, the system was able to differentiate between the actual characters and the dot (.) feature.

To find out that the dot feature belongs to which character, K-nearest neighbors (KNN) is used. KNN is a supervised machine learning algorithm used for classification and regression tasks but for this study, K-NN is used for classification as this is a classification problem. The main idea behind KNN is to predict the class or value of a new instance by comparing it to the k nearest neighbours in the given dataset. In other words, KNN determines the label of a new data point based on the labels of its neighbouring data points. The number of neighbours, k , is a hyper-parameter that needs to be specified before applying

the algorithm. For this study the value of K is 1 as dot feature can be a part of a character only.

4.3.2 Steps of KNN

4.3.2.1 Calculate distances

For each new instance in the testing dataset, compute the distance to every instance in the training dataset. In this study, the pixel values of each character and dot (.) feature were taken into account as the features. Five specific features were considered: top-right, bottom-right, top-left, bottom-left, and centre pixel values of each character and dot feature. The study utilized Euclidean distance formulas to calculate the distance between the characters and dot (.), which is repressed by Equation 4.10.

$$\text{Euclidean distance} = \sum_{i=1}^n \sqrt{(x_i - y_i)^2} \quad \text{Equation 4.10}$$

Here, x_i , y_i , and n represent the features of characters, dot (.), and the total of five features considered in the study, respectively.

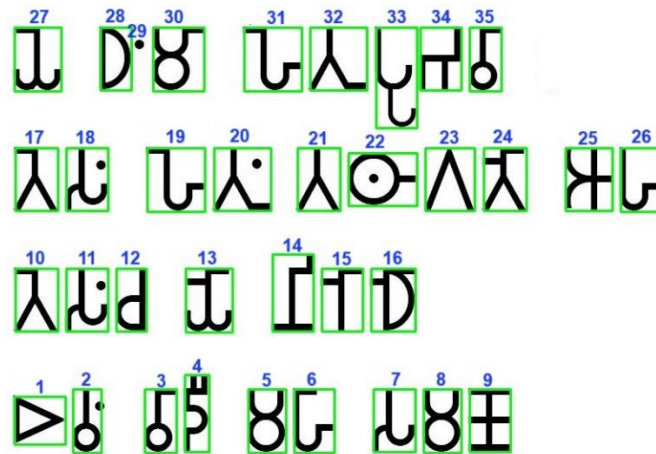


Figure 4.8: After merging the dot character with the relevant character

4.3.2.2 Classification

For classification tasks, determine the class label of the new instance based on the majority class of the k nearest neighbours. This is done by taking a vote among the class

labels of the neighbours. However, in this case, the dot (.) feature is merged with the nearest characters without requiring any voting process, as the value of K is set to 1. As illustrated in Figure 4.8, it is evident that all dot (.) attributes have been combined with the adjacent character on the left. The algorithm of K-NN is mentioned further.

Algorithm: K-Nearest Neighbours (KNN) with K=1 and Euclidean Distance

Input:

- Training dataset: X_train (array-like, shape [n_samples, 5]) - features of training instances
- y_train (array-like, shape [n_samples]) - class labels for training instances
- Testing instance: x_test (array-like, shape [5]) - features of the testing instance

Output:

- Predicted class label for the testing instance: y_pred
1. Calculate distances:
 - Initialize an empty list distances[]
 - for each instance x_train, y_train in zip(X_train, y_train) do a. Calculate the Euclidean distance dist between x_train and x_test:
 - $\text{dist} = \sqrt{\sum((x - y)^2 \text{ for } x, y \text{ in } \text{zip}(x_{\text{train}}, x_{\text{test}}))}$
 - Append (dist, y_train) to distances[] (pairing distance with corresponding label)
 2. Sort distances[] in ascending order based on distance values.
 3. Select the nearest neighbour:
 - Get the label of the nearest neighbour: $y_{\text{pred}} = \text{distances}[0][1]$
 4. Return the predicted class label y_pred

After applying the KNN, all dot (.) features merged with previous characters and the output of K-NN is represented by Figure 4.8.

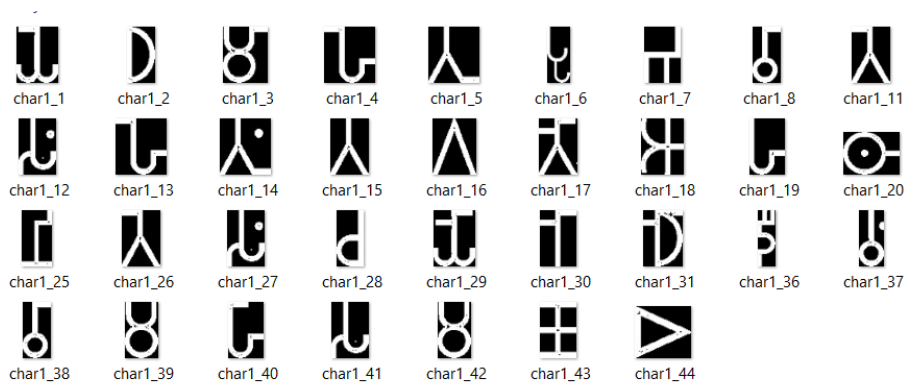


Figure 4.9: Example of all segmented character

All of the characters that have been defined by their ROI, are now regarded as individual characters and are saved. The segmented character can be seen in Figure 4.9.

4.4 Data Augmentation

Data augmentation, a fundamental concept in the realm of machine learning and data science, serves as a powerful technique for enhancing the quality, diversity, and quantity of datasets (Shorten et al., 2021). This approach plays a pivotal role in improving the robustness and generalization capabilities of machine learning models. In this context, the following sections delve into the methods, and applications of data augmentation, shedding light on its paramount role in the Brahmi samples and its recognition field. In data augmentation, "input space" and "feature space" refer to different levels at which data can be modified to create augmented versions for training machine learning models. Figure 4.10 mentions the taxonomy of data augmentation.

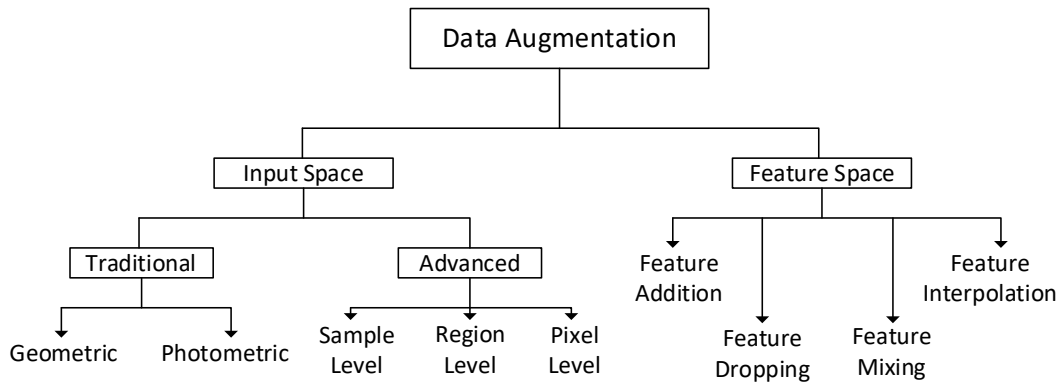


Figure 4.10: Hierarchy of the Data Augmentation

Table 4.2: Analysis of data augmentation techniques

Augmentation	Basic Technique	Comments
Geometric	Rotation, shearing, translation, scaling (excluding zooming or cropping), mirroring, and reflection, geometric changes	Each of these techniques has the potential to alter the positioning of features, particularly the "matra," resulting in changes to the meaning of compound characters. Therefore, these methods may not be appropriate for generating new samples within the context of the Brahmi script.
Photometric	Adjusted, Deformed, Blurred, Translated, and Noisy	These techniques focus on simulating various photometric effects caused by factors such as camera artifacts, shooting conditions, and image data corruption, and they alter the pixel values and appearance of images accordingly to enhance the dataset. It means that these techniques are neither changing position of any pixel nor removing features, object. Thus, it could be suitable to generate the samples of Brahmi script.
Sample Level	MixUp, CutMix	These data augmentation techniques generate new samples by combining two images, extracting a small portion from one image and inserting it into another, or removing the features. However, these methods may not be suitable for Brahmi scripts because the precise arrangement and combination of features play a crucial role in defining specific characters. Hence, both the features themselves and their positions are significant factors.
Region Level	Attention cropping, Object removal	
Pixel Level	Color Jittering, Image Inpainting	The entire system operates with black and white images. Additionally, it is important to note that altering a few pixels can have a substantial impact on the meaning of character within the Brahmi script. Consequently, techniques involving changes in color and inpainting may not be practical for generating new samples in the context of the Brahmi script.

Table 4.2 continued

Feature Addition	New features or dimensions	All of these data augmentation methods primarily center around features and modify them accordingly to create new samples. However, in the case of compound characters within the Brahmi script, altering individual elements could result in changes in meaning. Consequently, these techniques may not be well-suited for Brahmi script.
Feature Dropping	Removing or dropping certain features	
Feature Mixing	Combines two or more existing features within a dataset to create new composite features	
Feature Interpolation	Generating new feature vectors that lie between two or more existing feature vectors	

As indicated in Table 4.2, adjusting, deforming, blurring, translating, and adding noise to an image or dataset can be useful for the Brahmi characters. Therefore, these methods are categorized as part of data augmentation to enhance both the quality and quantity of the Brahmi samples. Here is a step-by-step methodology for each of these operations:

Image Adjustment
Input: Original images or dataset Output: Adjusted images Procedure: Adjust brightness, contrast, and saturation levels Apply color balance and tone mapping Normalize pixel values to a specified range [0, 1]
Image Deformation
Input: Original images or dataset Output: Deformed images Procedure: Apply random geometric transformations such as scaling, and shearing
Image Blurring
Input: Original images or dataset Output: Blurred images Procedure: Apply Gaussian, motion of blurring filters with random parameters Adjust the blur intensity
Image Translation
Input: Original images or dataset Output: Translated images Procedure: Randomly shift the image in both the horizontal and vertical directions Use padding or cropping to handle empty areas resulting from translation
Image Noise Addition
Input: Original images or dataset Output: Noisy images Procedure: Add random noise to the image like Gaussian noise, salt-and-pepper noise Adjust the noise standard deviation

Following the data augmentation phase, the sample size is expanded and consolidated in a single directory. Subsequently, all individual characters need to be categorized into distinct folders based on their similar features. While all samples are in image format, a recognition model can assist in character recognition. Hence, the subsequent stage of the system is dedicated to character recognition.

4.5 Two-Phase Enhanced Brahmi Recognition (TPEBR) Method

In the context of an OCR system, the workflow involves input, preprocessing, segmentation, and classification, with preprocessing and segmentation having been previously addressed in subsections 4.2 and 4.3, respectively. Therefore, this section exclusively concentrates on the classification aspect. As indicated by existing literature, a two-phase approach employing deep learning architectures may be well-suited for classification, and this will be explored in detail.

As Brahmi compound characters are a combination of vowels and consonants, the features of consonants and vowels (“matra”) can be observed from the compound characters. In Brahmi words, there are 368 different types of classes. Among these 368 classes, there are 10 different types of vowels, 33 different types of consonants, and 325 types of compound characters. In the current Brahmi word identification method, a Brahmi character, which might be a vowel, consonant, or compound character, must be compared to the other 368 characters in order to be recognised.

Based on the features of consonants and “matra”, this study proposed a process which is the combination of two phases to recognise the Brahmi words. The proposed approach of this study was inspired by two-phase approach by Das et al. (2015). In their suggested two-phase approach, the first phase was used to remove the "matra" from the compound characters of text and the second phase was used to recognise all characters and "matra." In

this study, a similar approach to Das et al. (2015) is utilized to recognize the Brahmi words. However, as “matra” cannot be separated from Brahmi words due to the position of “matra”, a modification of the approach by Das et al. (2015) is suggested in this study. In the proposed enhanced method (Two-Phase Enhanced Brahmi Recognition (TPEBR)), all features of vowels and consonants are classified first. Following this, in the second phase, all types of compound characters are recognised. The output is the combination of the first and second phases. The detail of each phase is outlined below.

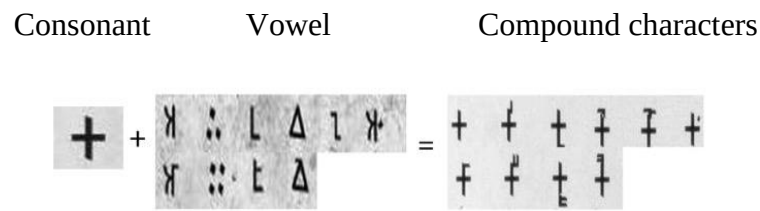


Figure 4.11: Consonant, Vowels, and Compound Characters of the Brahmi Script

First phase: All vowels and consonants are classified in this phase. Initially, all characters were divided into 43 groups, with 1 to 33 consonant groups and 34 to 43 vowel groups, according to 33 consonants and 10 vowel groups of the Brahmi script. Vowels and consonants can be easily categorised in this phase. Furthermore, because the compound characters consist of consonants, all compound characters created by the same consonant were kept in the same group after classification. For example, a consonant is followed by various vowels and creates several compound characters, and these compound characters belong to one group because all these compound characters were developed by the same consonant (Figure 4.11).

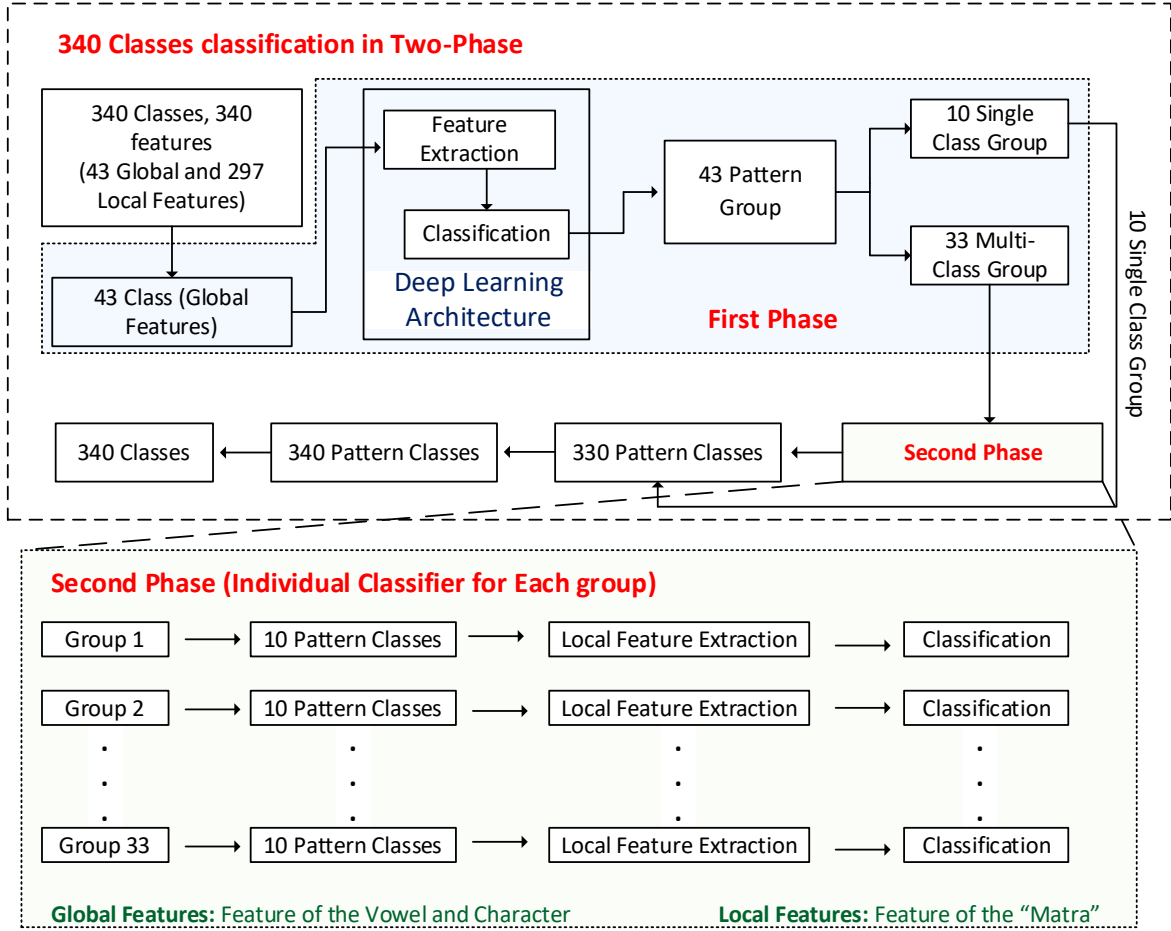


Figure 4.12: A Schematic Representation of Proposed Approach Two-Phase Enhanced Brahmi Recognition (TPEBR)

Second Phase: According to Das et al. (2015), a total of 33 classifiers are used to classify all the compound characters. During the training process of the classifier, only the compound characters with the same consonant were deemed to belong to the same group. Hence, with the 33 consonants, a total of thirty-three classifiers were trained. During the testing process in the second phase, each classifier will be used to recognise all the characters in that group. The proposed framework is presented in Figure 4.12 and Figure 4.13 explains further the process with an example.

Das et al. (2015) divided the text into 3 zones in the first phase, separated the “matra” from the text, and recognized all the content of the 3 zones in the second phase. The first and

third zone has “matra” and the second zone has the rest of the text. Different from this approach, the proposed enhanced approach (Two-Phase Enhanced Brahmi Recognition (TPEBR)) does not separate “matra” at any stage and this is the main difference in both approaches.

Within the Two-Phase Enhanced Brahmi Recognition (TPEBR) approach, a classification step is necessary to categorize the characters. As emphasized in existing literature, deep learning architectures are preferred for classifying Brahmi characters and compound characters. However, it is important to note that deep learning encompasses various architecture types, each with its own specific input image size requirements. Consequently, prior to applying deep learning, image resizing becomes a crucial step.

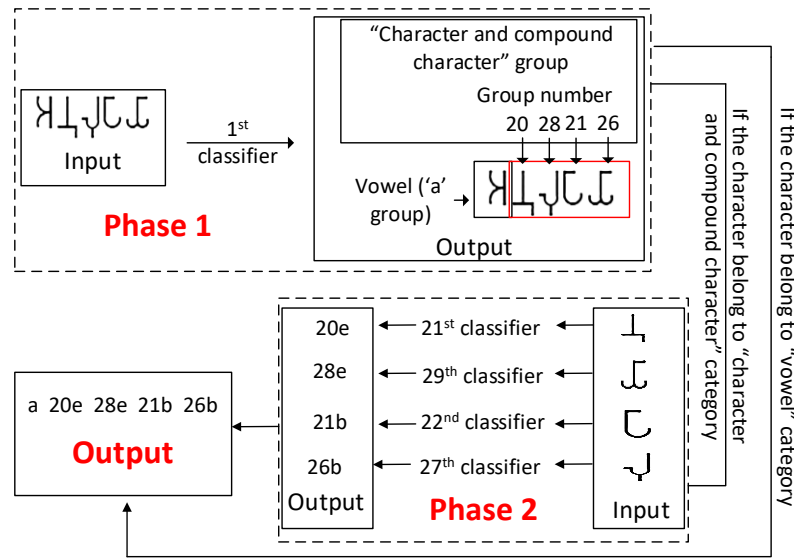


Figure 4.13: Process of Proposed Approach (Two-Phase Enhanced Brahmi Recognition (TPEBR)) with an Example

The entirety of the recognition process is divided into two sequential steps. Initially, the approach to resizing is discussed, followed by name of deep learning architectures is provided.

4.5.1 Size Normalization (Resizing)

Normalisation was used to make all of the characters the same size. Because the characters differed in size, their sizes should be made consistent to allow for comparison. Normalisation should reduce or increase the original size of the image without altering the structure of an image.

The thresholding (Section 4.2.1.2, part I) indicated that the pixel value of the image was either zero or one. Therefore, resizing was straightforward (Yousaf et al., 2019). The image resizing was performed using bi-linear interpolation (Michalak & Okarma, 2019), which works in both directions (x, y) and at one pixel at a time. Assuming that x_k, y_k is a two-dimensional (2D) image and if $k = 1, 2$, these coordinates define the position of points Q_{11}, Q_{21}, Q_{12} , and Q_{22} . Applying bi-linear interpolation for a point P between Q_{11}, Q_{21}, Q_{12} , and Q_{22} was calculated using the following equation:

$$f(x, y_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})$$

$$f(x, y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}),$$

where the P value coordinates were derived in the following equation:

$$f(x, y) \approx \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2)$$

$$= \frac{y_2 - y}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \right)$$

$$+ \frac{y - y_1}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \right)$$

$$= \frac{1}{(x_2 - x_1)(y_2 - y_1)} [x_2 - x \quad x - x_1] \begin{bmatrix} f(Q_{11}) & f(Q_{12}) \\ f(Q_{21}) & f(Q_{22}) \end{bmatrix} \begin{bmatrix} y_2 & y \\ y & y_1 \end{bmatrix},$$

which would yield the result of bi-linear interpolation on an image.

After applying bi-linear interpolation to resize the all samples, mean, median, mode, min, max value of Q_{11} , Q_{21} , Q_{12} , and Q_{22} is presneted in Table 4.3.

Table 4.3: Statistics Value of Q_{11} , Q_{21} , Q_{12} , and Q_{22}

	Mean	Median	Mode	Min	Max
Q_{11}	172	127.0	255	0	255
Q_{21}	171	127.0	255	0	255
Q_{12}	168	127.0	255	0	255
Q_{22}	168	127.0	255	0	255

4.5.2 Architecture

This research explored a variety of deep learning architectures, including AlexNet (Krizhevsky et al., 2017), DenseNet121 (Huang et al., 2017), DenseNet169 (Huang et al., 2017), DenseNet201 (Huang et al., 2017), Inceptionv3 (Szegedy et al., 2016), MobileNet (Phiphaphatphaisit & Surinta, 2020), MobileNetV2 (Sandler et al., 2018), MobileNetV3_Large (Howard et al., 2019), MobileNetV3_Small (Howard et al., 2019), ResNet50 (He et al., 2016), ResNet50V2 (Qazi et al., 2022), ResNet101 (He et al., 2016), ResNet101V2 (Bilal et al., 2022), ResNet152 (He et al., 2016), ResNet152V2 (Bilal et al., 2022), , InceptionResNetV2 (Szegedy et al., 2017), ResNeXt (Xie et al., 2017), SqueezeNet (Iandola et al., 2016), VGG16 (Simonyan & Zisserman, 2014), VGG19 (Simonyan & Zisserman, 2014), Xception (Yao et al., 2021). These architectures were considered for the classification phase within the proposed two-phase approach. Input size of each classifier is presented in Table 4.4.

Following the recognition step, where all characters have been successfully identified, the subsequent task involves grouping similar characters into designated folders. In the next phase, the identified characters can be systematically organized into their respective folders automatically.

Table 4.4: Standard Size of each Architecture

Architetur	Size	Architetur	Size
AlexNet	227x227x3	ResNet152	224x224x3
DenseNet121	224x224x3	ResNet152V2	224x224x3
DenseNet169	224x224x3	ResNet50	224x224x3
DenseNet201	224x224x3	ResNet50V2	224x224x3
InceptionV3	299x299x3	InceptionResNetV2	299x299x3
MobileNet	224x224x3	ResNeXt	224x224x3
MobileNetV2	224x224x3	SqueezeNet	224x224x3
MobileNetV3_Large	224x224x3	VGG16	224x224x3
MobileNetV3_Small	224x224x3	VGG19	299x299x3
ResNet101	224x224x3	Xception	224x224x3
ResNet101V2	224x224x3		

4.6 Data Storage and Labelling

All individual characters were standardized to a consistent format, necessitating the conversion of all extracted character images into JPG format. In the recognition phase, the arrangement and categorization of characters were determined, and an automated process deposited them into designated folders. Every folder is assigned a distinct name that corresponds to groups of characters with similar characteristics, a process facilitated by the recognition stage.

**Figure 4.14:** All Isolate Characters are Stored into Groups by using Segmentation and Recognition Step

For instance, the characters extracted from the Brahmi text in Figure 4.14 underwent isolation using the segmentation approach discussed in subsection 4.1.3. Subsequently, they underwent recognition through the procedure outlined in subsection 4.1.5. The recognized characters were then organized into distinct groups, with each group containing a single type of character or compound character. The objective was to create a total of 368 groups, but resource limitations led to the creation of 170 classes (groups) encompassing 27 variations of consonants, 4 types of vowels, and the remaining 129 representing compound characters.

Following that, the labelling phase is carried out. Unicode serves as the labelling method (Kushwaha & Singh, 2020). Each folder adopts the name of a Brahmi character that corresponds to its respective class. To improve clarity and visual representation, folder names are substituted with codes, with each code symbolizing a particular Brahmi character. For a visual representation, Figure 4.15 displays the labeling of a few folders within the Brahmi dataset.



Figure 4.15: Labeling of few Samples of Brahmi Dataset

It is important to note that this labelling approach does not involve any additional techniques or complexity. The performance of the Brahmi recognition model remains unaffected by this automatic labelling process, as it solely focuses on mapping the output classes to their corresponding folders for organization and clarity purposes.

In the last, flow of entire methodology of proposed system to do automatic labelling along with training and testing details is presented by Figure 4.16.

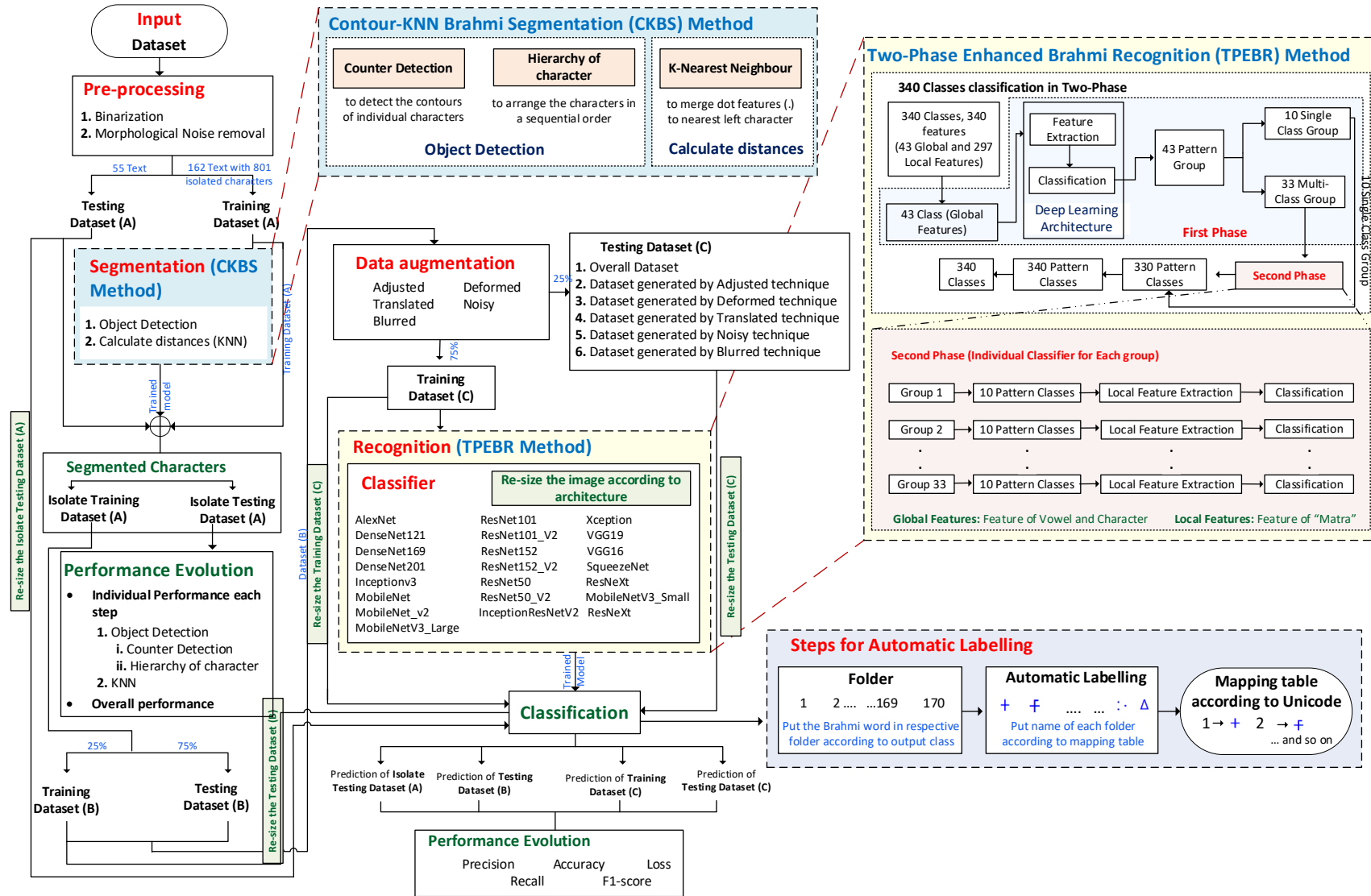


Figure 4.16: Flow of Methodology of Proposed System to do Automatic Labelling along with Training and Testing Details

4.7 Summary

This chapter is primarily devoted to the implementation of an automated system designed for the creation of a Brahmi dataset. The initial phase within this system revolves around collecting samples, serving as the foundational step in dataset generation. Although Brahmi script samples were gathered from various online sources, it was evident that these samples often lacked uniformity in text and word sizes. Consequently, the subsequent phase involved isolating characters from words and text, employing the proposed segmentation approach (hybrid approach of object detection and KNN). This step, referred to as the segmentation stage, is pivotal and precedes system training.

Furthermore, the quality and quantity of dataset were enriched through the application of data augmentation techniques, encompassing adjustments, deformations, blurring, translations, and the introduction of noise. Following this, all the collected samples underwent recognition using diverse deep learning architectures with proposed two-phase approach. It is worth noting that each deep learning architecture has distinct input size requirements, necessitating resizing of the samples to match the specified dimensions prior to the recognition stage. The Proposed approach followed a two-phase approach, with the initial and subsequent phases dedicated to examining global characteristics (vowel and consonant features) and local features (known as 'matra'), respectively. In the last, all recognized characters were automatically categorized into folders, and folder labelling was executed through Unicode, with each unique Unicode code corresponding to a specific Brahmi character.

CHAPTER 5

EXPERIMENTAL RESULTS

In this chapter, the assessment of the proposed Brahmi dataset system is presented, encompassing a series of steps, including input image acquisition, pre-processing, segmentation, data augmentation, recognition, data storage, and labeling. It is important to note that the process of gathering input images from diverse online sources was executed meticulously to ensure that the performance of system remained unaffected.

For a more comprehensive analysis of the effectiveness of system, it has been divided into two distinct parts. The first part of the evaluation primarily focuses on the segmentation step (Contour-KNN Brahmi Segmentation (CKBS)). Additionally, the significance of data augmentation in the recognition phase is acknowledged. However, the augmentation approach to build the training and testing dataset thus, it is assume that augmentation approach did not affect the performance. Therefore, only Two-Phase Enhanced Brahmi Recognition (TPEBR) was used for assessment in the second part of the evaluation. Ultimately, the overall performance of the entire system is determined by amalgamating the assessment of both parts. In the last, a performance comparison with the accuracies of previous studies is conducted. Figure 5.1 presents the flow of performance evaluation of proposed system.

The initial stage in developing the Brahmi dataset system involves data collection from various websites. This study gathered a total of 217 samples comprising text and words, with non-uniform sizes, indicating variations in characters within words and words within text. Furthermore, the study collected an additional 801 samples of Brahmi characters and compound characters that had already been isolated.

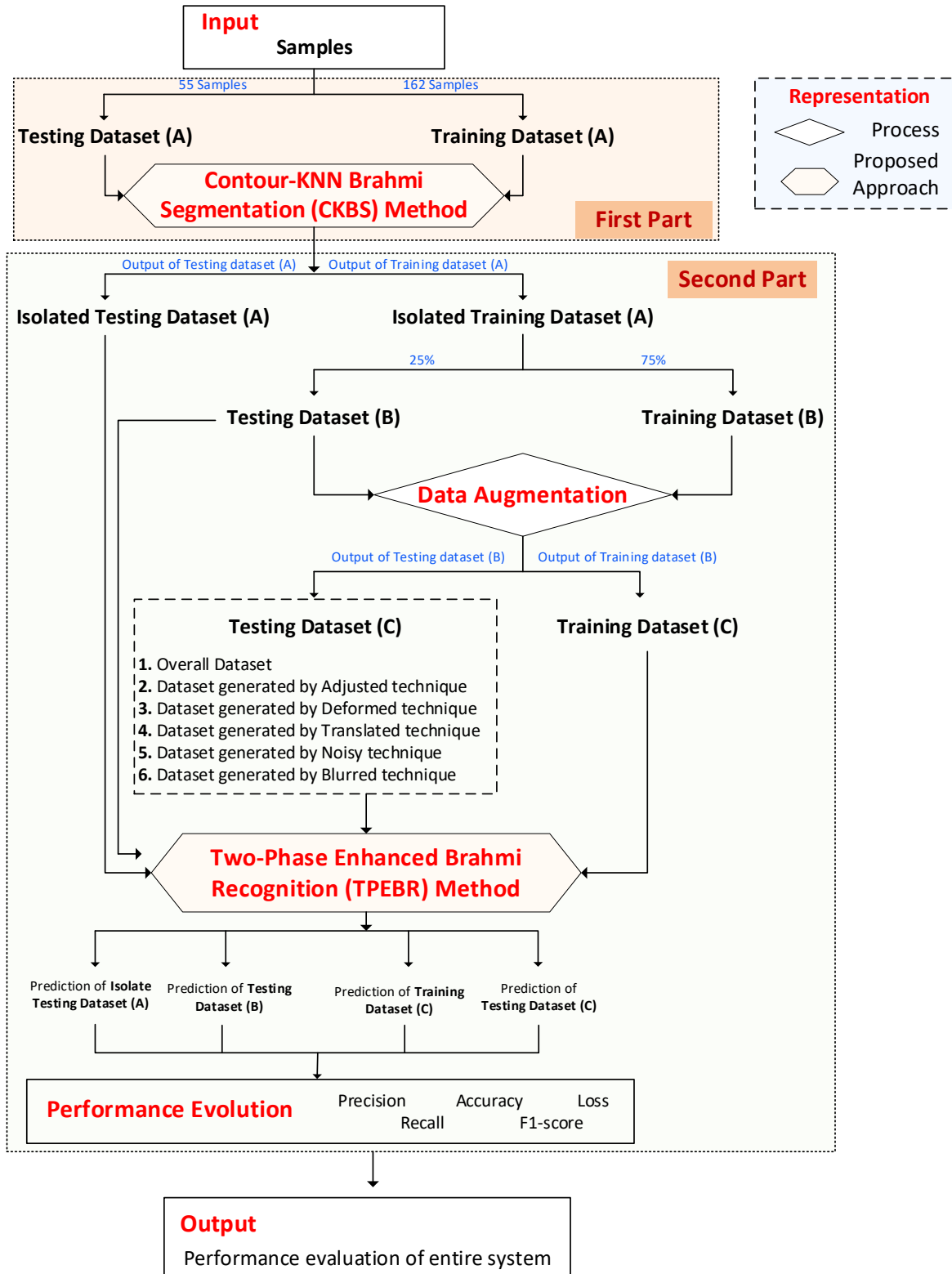


Figure 5.1: Flow of the Performance Evaluation of the Proposed System

5.1 Result Contour-KNN Brahmi Segmentation (CKBS) (First Part)

The methodology employed in this study consisted of two parts: object detection and KNN, which were executed sequentially. The Brahmi dataset, consisting of 162 text samples for training and 55 for testing, was employed. Additionally, a total of 801 segmented characters were included. Among these characters, there were 49 complex compound characters and 250 characters with dot (.) features. The performance of the object detection approach was assessed solely on characters without dot (.) features due to the limitations of the object detection part. Out of the 551 characters tested (total 801 character-250 characters with dot (.) features), 550 were successfully segmented, except for Λ complex compound characters. These complex compound characters were treated as two separate objects by the approach, leading to incorrect segmentation. In the text, one instance of this character was correctly segmented out of two appearances. Additionally, out of the 49 complex compound characters in the testing dataset, 48 were segmented accurately. It can be concluded that the object detection approach struggled to recognize the connecting line in Λ complex compound characters, resulting in them being treated as separate entities.

Furthermore, the method focuses on evaluating the performance of KNN in segmenting characters with dot (.) features. The KNN model was trained only on characters with dot (.) features. For training and testing the KNN model, 175 and 75 samples of characters with dot (.) features were used, respectively. In the testing phase, the system correctly recognized 73 out of 75 samples with dot (.) features, which were unseen data for the KNN model. The performance of KNN to segment the Brahmi character with dot (.) feature was 97.33% which is calculated by Equation 5.1.

After evaluating the performance of each model separately, both approaches were combined and tested on 55 text samples containing a total of 663 characters. The system

incorrectly segmented 1 character in the first stage and 11 characters in the second stage. Therefore, a total of 12 characters out of 663 were segmented incorrectly. The proposed approach successfully segmented 651 characters, achieving an accuracy rate of 98.19%. The accuracy of the system was calculated using Equation 5.1.

$$Accuracy = \frac{\text{Number of correctly segmented character}}{\text{Total number of characters}} \times 100 \quad \text{Equation 5.1}$$

In addition to testing the proposed approach on Brahmi text, this study also incorporated the use of Tamil script, which shares similar features with the Brahmi script as Tamil script originated from Brahmi script.

To evaluate the effectiveness of the proposed approach in segmenting characters from Tamil script, the PHDIndic_11 dataset (Obaidullah et al., 2018), consisting of handwritten Tamil text samples, was employed. The dataset contained a total of 120 text samples. From these samples, 20 images were selected for analysis, with 8 images specifically used for testing the object detection part. Additionally, characters with dot (.) features were extracted from all 10 images, while the remaining 10 samples were utilized to assess the performance of the proposed system. The testing dataset comprised 536 characters with dot (.) features and 1,082 characters without dot (.) features, enabling separate evaluation of the object detection and KNN components.

The object detection approach was first tested on the 1,082 samples without dot (.) features, resulting in 997 correctly segmented characters. Similarly, the performance of the KNN model was assessed on the 536 samples with dot (.) features, and it successfully segmented 509 characters. The respective accuracies of the object detection and KNN were determined to be 92.14% and 96.82%. To evaluate the overall performance of the proposed model, 10 different Tamil texts were used, consisting of a total of 1,482 characters. Among

these, 1,337 characters were correctly segmented, yielding an accuracy of 90.21% for the proposed approach in segmenting Tamil characters from the text.

5.2 Data Augmentation

Data augmentation was applied only on isolated characters of the training dataset of the previous step. After applying segmentation approach to 162 text, various samples were collected from 170 classes however, number of samples of each class is not uniform and it may cause bias issues. So, this study used 40 samples of each class where 30 and 10 for training and testing, respectively. Initially, the samples were randomly organized, with 30 samples in each class for the training dataset and 10 samples for the testing dataset.

To enhance the size and diversity of the dataset, a data augmentation technique was applied. This involved adjusting, blurring (Gaussian blur), deforming, and introducing noise (salt and pepper noise, Gaussian noise) and translated. As a result, a total of 150 samples, 50 for each class, were generated for the training dataset. Additionally, 50 samples, 10 for each class, were created for the testing dataset, utilizing the same augmentation techniques. After augmentation, the dataset expanded to include 180 samples per class for the training dataset and 60 samples per class for the testing dataset.

5.3 Result of Two-Phase Enhanced Brahmi Recognition (TPEBR) Method (Second part)

The performance of the TPEBR method was evaluated on the Brahmi dataset where 180 and 60 samples of each class for a training and testing dataset, respectively. It means that 30,600 and 10,200 are the total number of samples for training and testing, respectively.

5.3.1 Performance Metrics

Different parameters can be employed to evaluate the performance of a system such as parameters include loss function, precision & recall, F1-score, specificity or True Negative Rate (TNR), and accuracy.

5.3.1.1 Loss Function: The specific formula for the loss function depends on the type of machine learning problem like working on regression, classification. This study used mean squared error.

The Mean Squared Error (MSE) is a common loss function to measure the average squared difference between predicted values and actual values. The formula for Mean Squared Error is as follows:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad \text{Equation 5.2}$$

Where n , y_i , $\hat{y}_i$ represents the total number of data points, the actual (ground truth) value of the target for the i^{th} data point and the predicted value of the target for the i^{th} data point, respectively.

5.3.1.2 Precision & Recall: Before calculating precision and recall, it is essential to determine the values of TP, FN, FP, and TN. Recall, which measures the ability of the system to identify relevant instances, is computed by dividing true positives by the sum of true positives and false negatives. Similarly, precision, which measures the ability of the system to identify only relevant instances, is calculated by dividing true positives by the sum of true positives and false positives. Equations 5.3 and 5.4 show the formulas for precision and recall, respectively.

$$Precision = \frac{TP}{TP+FP} \quad \text{Equation 5.3}$$

$$Recall = \frac{TP}{TP+FN} \quad \text{Equation 5.4}$$

5.3.1.3 F1-Score: The F1 score is a metric that takes into account both precision and recall. It is calculated by taking the harmonic mean of precision and recall, which is the multiplication of precision and recall values, divided by the sum of precision and recall values. The formula for the F1 score is given by equation 5.5.

$$F_1 = \frac{2 * precision * recall}{Precision + Recall} \quad \text{Equation 5.5}$$

5.3.1.4 Specificity or True Negative Rate (TNR): The True Negative Rate (TNR) is a measure that is used to determine the actual negative values. It is calculated by taking the ratio of true negative to the addition of false positive and true negative values. This measure can be represented by the equation 5.6.

$$True\ Negative\ Rate\ (TNR) = \frac{TN}{FP + TN} \quad \text{Equation 5.6}$$

5.3.1.5 Accuracy: The most widely used and straightforward parameter for evaluating the efficiency of any system is accuracy. It is determined by dividing the number of correctly identified samples by the total number of samples, and then multiplying by 100. The formula for accuracy is represented by equation 5.7.

$$Accuracy = \frac{TP + FN}{TP + FP + TN + FN} * 100 \quad \text{Equation 5.7}$$

5.3.2 Result of the Recognition Model

When analyzing the performance of different architectures across various metrics, it becomes evident that certain models excel in specific categories, while others struggle to maintain a competitive edge. This evaluation allows to identify the best and worst models within each category and to determine an overall ranking.

Among the architectures where the proposed approach was used, SqueezeNet stands out with the lowest loss of 0.2897, showcasing its ability to minimize errors and discrepancies in predictions. In contrast, ResNeXt exhibits the highest loss of 4.9130, indicating that it struggles in this aspect (Table 5.1).

Table 5.1: Performance of TPEBR Approach where Acc, Pre, Rec, F1 Represents Accuracy, Precision, Recall and F1-Score

Architectures	Loss	Acc	Pre	Rec	F1
AlexNet	2.3461	84.37	77	84	80
DenseNet121	0.5982	90.23	92	90	90
DenseNet169	0.4771	88.76	89	89	88
DenseNet201	0.4325	89.86	90	90	89
Inceptionv3	0.4759	88.72	89	89	88
MobileNet	0.4586	94.15	94	94	93
MobileNetV2	0.5286	94.85	91	91	90
MobileNetV3_Large	0.7424	83.67	86	84	83
MobileNetV3_Small	1.4221	65.94	68	66	63
ResNet101	1.4576	56.50	60	56	53
ResNet101V2	0.4843	87.27	88	87	87
ResNet152	1.4321	61.19	64	61	57
ResNet152V2	0.5223	86.98	87	87	86
ResNet50	1.7997	48.65	50	49	43
ResNet50V2	0.4338	90.15	90	90	89
InceptionResNetV2	0.5118	87.80	88	88	87
ResNeXt	4.9130	03.04	00	03	01
SqueezeNet	0.2897	97.58	96	96	96
VGG16	1.9482	45.02	44	45	38
VGG19	2.6445	30.48	33	30	25
Xception	0.7871	77.45	81	77	76

SqueezeNet emerges as the leader in accuracy, boasting an impressive 97.58%, making it the most proficient model in accurately classifying data. On the other hand, ResNeXt lags with a significantly second-lowest accuracy of 03.04%.

The performance of the Two-Phase Enhanced Brahmi Recognition (TPEBR) Method was assessed across a range of datasets, encompassing variations such as adjustments, blurring, deformations, noise, translations, the original dataset, and the testing dataset from the segmentation part.

Table 5.2: Performance of TPEBR Approach on Adjusted Data Augmentation Testing Sample Where Acc Represents the Accuracy

Architectures	Loss	Acc	Precision	Recall	F1-score
AlexNet	2.4674	84.11	77	84	79
DenseNet121	0.6576	89.89	91	90	89
DenseNet169	0.5056	89.13	88	89	88
DenseNet201	0.4837	89.13	89	89	88
Inceptionv3	0.480	89.99	90	90	89
MobileNet	0.5195	93.95	93	94	93
MobileNetV2	0.5352	92.92	92	93	92
MobileNetV3 Large	0.7333	87.34	88	87	86
MobileNetV3 Small	1.4989	69.29	71	69	66
ResNet101	1.4147	58.47	59	58	54
ResNet101_V2	0.4881	87.93	89	88	87
ResNet152	1.3603	63.66	62	64	58
ResNet152_V2	0.5249	88.42	88	88	87
ResNet50	1.6518	52.11	49	52	45
ResNet50_V2	0.4754	90.72	91	91	90
InceptionResNetV2	0.5274	89.20	89	89	88
ResNeXt	4.9938	03.01	00	03	01
SqueezeNet	0.3111	95.86	95	96	95
VGG16	1.9129	48.68	46	49	42
VGG19	2.6380	31.54	32	32	26
Xception	0.7666	79.02	81	79	77

In the conclusion of all tables (Table 5.2 to 5.8), the "ResNeXt" architecture is the worst model, unfortunately, demonstrates the weakest performance in this analysis. With a remarkably low accuracy of less than 4%, combined with precision, recall, and F1-score values between zero to four, it indicates a significant struggle in making accurate predictions.

Among the models evaluated using the TPEBR Approach, "SqueezeNet" emerges as the top-performing architecture. With a remarkable accuracy of more than 95%, SqueezeNet showcases an excellent balance of precision, recall, and F1-score. This suggests that the Proposed Approach significantly enhances ability of SqueezeNet to recognize Brahmi words.

Table 5.3: Performance of TPEBR Approach on Blurred Data Augmentation Testing Sample where Acc Represents the Accuracy

Architectures	Loss	Acc	Precision	Recall	F1-score
AlexNet	2.1286	84.70	77	85	80
DenseNet121	0.6018	91.20	92	91	90
DenseNet169	0.5352	88.46	88	88	87
DenseNet201	0.4695	90.04	90	90	89
Inceptionv3	0.4677	90.92	91	91	90
MobileNet	0.5947	93.85	93	94	93
MobileNetV2	0.6138	90.36	90	90	89
MobileNetV3 Large	1.1857	73.14	80	73	72
MobileNetV3 Small	1.6820	60.34	65	60	58
ResNet101	1.8618	49.08	54	49	46
ResNet101_V2	0.5235	86.97	87	87	86
ResNet152	1.8236	54.33	56	54	50
ResNet152_V2	0.5587	86.77	87	87	85
ResNet50	2.4014	37.87	38	38	33
ResNet50_V2	0.4882	90.29	90	90	89
InceptionResNetV2	0.5451	88.39	89	88	87
ResNeXt	4.9104	03.58	00	04	01
SqueezeNet	0.2978	96.05	95	96	95
VGG16	2.1521	43.37	38	43	36
VGG19	2.8694	29.09	26	29	23
Xception	7907	80.47	83	80	79

Among the models evaluated using the TPEBR Approach, "SqueezeNet" emerges as the top-performing architecture. With a remarkable accuracy of more than 95%, SqueezeNet showcases an excellent balance of precision, recall, and F1-score. This suggests that the Proposed Approach significantly enhances ability of SqueezeNet to recognize Brahmi words.

Table 5.4: Performance of TPEBR Approach on Deformed Data augmentation Testing Sample

Architectures	Loss	Accuracy	Precision	Recall	F1-score
AlexNet	2.3595	84.16	77	84	79
DenseNet121	0.8223	87.69	90	88	87
DenseNet169	0.5680	87.15	87	87	86
DenseNet201	0.5407	88.04	88	88	87
Inceptionv3	0.6156	85.27	86	85	85
MobileNet	0.5361	93.25	93	93	92
MobileNetV2	0.6529	87.69	88	88	86
MobileNetV3 Large	0.8399	84.16	85	84	83
MobileNetV3 Small	1.6206	64.59	65	65	61
ResNet101	1.5428	55.94	58	56	51
ResNet101_V2	0.6011	85.74	87	86	85
ResNet152	1.5115	60.66	63	61	56
ResNet152_V2	0.6458	84.33	86	84	83
ResNet50	1.8330	49.32	51	49	44
ResNet50_V2	0.5355	88.66	89	89	88
InceptionResNetV2	0.6800	84.04	85	84	83
ResNeXt	4.9948	03.26	00	03	01
SqueezeNet	0.3305	95.60	95	96	95
VGG16	2.1529	39.02	37	39	33
VGG19	2.6999	30.57	28	31	24
Xception	0.9373	73.93	75	74	72

Table 5.5: Performance of TPEBR Approach on Noisy Data Augmentation Testing Sample

Architectures	Loss	Accuracy	Precision	Recall	F1-score
AlexNet	2.3856	84.12	77	84	79
DenseNet121	0.6800	90.35	91	90	90
DenseNet169	0.5724	87.67	87	88	86
DenseNet201	0.4779	90.56	91	91	90
Inceptionv3	0.6013	85.26	87	85	84
MobileNet	0.5889	93.53	93	94	93
MobileNetV2	0.6728	88.83	89	89	88
MobileNetV3 Large	0.7787	86.17	87	86	85
MobileNetV3 Small	1.5293	66.05	69	66	62
ResNet101	1.4312	59.39	62	59	55
ResNet101_V2	0.6262	84.56	86	85	83
ResNet152	1.3748	63.13	63	63	58
ResNet152_V2	0.6536	85.69	86	86	84
ResNet50	1.7022	51.35	50	51	45
ResNet50_V2	0.4971	88.75	89	89	88
InceptionResNetV2	0.6681	85.91	87	86	85
ResNeXt	5.0762	02.90	00	03	01
SqueezeNet	0.3242	95.70	95	96	95
VGG16	1.9586	43.77	40	44	37
VGG19	2.6422	29.10	27	29	24
Xception	0.9253	73.91	76	74	71

Table 5.6: Performance of TPEBR Approach on Translated Data Augmentation Testing Sample

Architectures	Loss	Accuracy	Precision	Recall	F1-score
AlexNet	2.4895	83.67	77	84	79
DenseNet121	0.7752	87.39	88	87	86
DenseNet169	0.6041	87.88	88	88	87
DenseNet201	0.6170	87.10	88	87	86
Inceptionv3	0.6473	85.64	86	86	84
MobileNet	0.6749	92.97	92	93	92
MobileNetV2	0.6569	91.19	90	91	90
MobileNetV3 Large	0.9438	83.67	86	84	82
MobileNetV3 Small	1.8089	64.32	63	64	60
ResNet101	1.5366	60.53	55	61	55
ResNet101_V2	0.6719	81.14	81	81	79
ResNet152	1.5094	60.14	61	60	56
ResNet152_V2	0.6856	84.58	86	85	83
ResNet50	1.6950	53.73	47	54	46
ResNet50_V2	0.6633	86.03	87	86	85
InceptionResNetV2	0.7422	85.03	85	85	83
ResNeXt	5.2474	03.31	01	03	01
SqueezeNet	0.4378	92.74	92	93	92
VGG16	2.0601	45.76	39	46	38
VGG19	2.7796	27.74	24	28	22
Xception	0.9289	73.82	76	74	72

Table 5.7: Performance of TPEBR Approach on Original Data Testing Sample

Architectures	Loss	Accuracy	Precision	Recall	F1-score
AlexNet	2.2475	84.79	78	85	80
DenseNet121	0.3929	94.69	95	95	94
DenseNet169	0.3794	93.09	94	93	92
DenseNet201	0.3448	94.13	94	94	94
Inceptionv3	0.3541	94.52	94	95	94
MobileNet	0.3610	96.15	96	96	95
MobileNetV2	0.3482	95.06	95	95	95
MobileNetV3 Large	0.5993	87.20	89	87	86
MobileNetV3 Small	1.1920	71.01	75	71	68
ResNet101	1.2564	61.23	66	61	57
ResNet101_V2	0.3854	93.02	93	93	92
ResNet152	1.1720	65.49	68	65	61
ResNet152_V2	0.3909	91.90	92	92	91
ResNet50	1.6456	52.54	54	53	47
ResNet50_V2	0.3185	94.40	95	94	94
InceptionResNetV2	0.3883	93.82	94	94	93
ResNeXt	5.0673	02.99	00	03	01
SqueezeNet	0.1843	96.96	97	97	96
VGG16	1.8620	48.60	48	49	42
VGG19	2.5839	32.34	35	32	26
Xception	0.6565	84.38	87	84	84

Table 5.8: Comparison of Training and Testing Accuracy of TPEBR Approach where Acc represents Accuracy

Architectures	Training Accuracy					Testing Accuracy				
	Loss	Acc	Precision	Recall	F1-score	Loss	Acc	Precision	Recall	F1-score
AlexNet	2.0859	86.34	80	86	82	2.3461	84.37	77	84	80
DenseNet121	0.0982	97.79	98	98	98	0.5982	90.23	92	90	90
DenseNet169	0.2065	96.00	97	96	96	0.4771	88.76	89	89	88
DenseNet201	0.1764	97.60	98	98	98	0.4325	89.86	90	90	89
Inceptionv3	0.1870	97.06	97	97	97	0.4759	88.72	89	89	88
MobileNet	0.0935	98.40	99	98	98	0.4586	94.15	94	94	93
MobileNetV2	0.1006	97.45	98	97	97	0.5286	94.85	91	91	90
MobileNetV3 Large	0.4432	86.44	90	86	86	0.7424	83.67	86	84	83
MobileNetV3 Small	0.9353	71.56	75	72	70	1.4221	65.94	68	66	63
ResNet101	1.1463	60.65	65	61	57	1.4576	56.50	60	56	53
ResNet101_V2	0.2320	95.19	96	95	95	0.4843	87.27	88	87	87
ResNet152	1.0902	64.57	69	65	61	1.4321	61.19	64	61	57
ResNet152_V2	0.2369	94.39	95	94	94	0.5223	86.98	87	87	86
ResNet50	1.5960	52.25	59	52	47	1.7997	48.65	50	49	43
ResNet50_V2	0.1310	97.40	98	97	97	0.4338	90.15	90	90	89
InceptionResNetV2	0.1691	96.03	96	96	96	0.5118	87.80	88	88	87
ResNeXt	4.8961	03.07	00	03	01	4.9130	03.04	00	03	01
SqueezeNet	0.0759	98.96	99	99	99	0.2897	97.58	96	96	96
VGG16	1.7910	48.27	52	48	42	1.9482	45.02	44	45	38
VGG19	2.5077	33.17	39	33	27	2.6445	30.48	33	30	25
Xception	0.5534	86.25	89	86	86	0.7871	77.45	81	77	76

Finally, TPEBR approaches were evaluated using the same dataset that was initially used for the testing of segmentation approach. The results of this evaluation are presented in Table 5.9. This dataset consisted of 55 text samples, containing a total of 663 characters, with approximately 4 samples for each character class.

In the TPEBR approach, SqueezeNet emerges as the best model. It exhibits a minimal loss of 0.237 and an outstanding accuracy of 97.27%. This model also excels in precision, recall, and F1-score, showcasing its robust performance. On the other hand, ResNeXt Small performs relatively poorly in this category, with a higher loss of 4.9902 and accuracy at 3.015%. While not the weakest overall, it is one of the models that could benefit from further optimization.

Table 5.9: The performance TPEBR Approaches of the Testing Dataset of Segmentation Approach

Architectures	Loss	Acc	Precision	Recall	F1-score
AlexNet	2.2968	84.58	78	85	80
DenseNet121	0.4956	92.46	94	93	92
DenseNet169	0.4283	90.93	92	91	90
DenseNet201	0.3887	92.00	92	92	92
Inceptionv3	0.4150	91.62	92	92	91
MobileNet	0.4098	95.15	95	95	94
MobileNetV2	0.4384	93.05	93	93	93
MobileNetV3 Large	0.6709	85.44	88	86	85
MobileNetV3 Small	1.3071	68.48	72	69	66
ResNet101	1.357	58.87	63	59	55
ResNet101_V2	0.4349	90.15	91	90	90
ResNet152	1.3021	63.34	66	63	59
ResNet152_V2	0.4566	89.44	90	90	89
ResNet50	1.7227	50.60	52	51	45
ResNet50_V2	0.3762	92.28	93	92	92
InceptionResNetV2	0.4501	90.81	91	91	90
ResNeXt	4.9902	3.015	0	3	1
SqueezeNet	0.2370	97.27	97	97	96
VGG16	1.9051	46.81	46	47	40
VGG19	2.6142	31.41	34	31	26
Xception	0.7218	80.92	84	81	80

In the overall assessment, the proposed approach appears to yield good results. SqueezeNet in the TPEBR approach takes the crown as the best-performing model, displaying excellent performance across multiple metrics. In contrast, the ResNeXt architecture consistently struggles in both approaches, establishing itself as the weakest model overall. It is evident that the proposed approach brings notable enhancements to the performance of various architectures, underlining its potential in optimizing deep learning models.

5.3.3 Comparison the performance of Two-Phase Enhanced Brahmi Recognition (TPEBR) Method with Previous Studies

The performance comparison between the existing and proposed Brahmi word recognition systems was presented in Table 5.13, where the TPEBR approach performed better than the existing systems. In previous studies, systems that used existing OCR systems with one classifier (single-phase approach) and local datasets to build a Brahmi word recognition system were suggested. The proposed Brahmi word recognition model used the two-phase approach and a proposed dataset to introduce a Brahmi word recognition system.

Table 5.10: Comparison of Existing and Proposed Brahmi Word Recognition Systems

Brahmi word recognition systems	Accuracy
Soumya and Kumar (2015)	80.20%
Vellingiriraj et al. (2016)	90.24%
Proposed System	97.58%

Soumya and Kumar (2015) and Vellingiriraj et al. (2016) used Gabor features and zoning method+ ANN and zonal density features + ANN, respectively, to recognise Brahmi words with a traditional Process and achieved 80.20% and 90.24% accuracy, respectively. Furthermore, the proposed Process used the SqueezeNet classifier to recognise Brahmi words and achieved 97.58% accuracy, which was better than that of previous studies.

5.4 Overall Performance of Proposed System

All elements of the proposed system, encompassing input handling, pre-processing, segmentation, data augmentation, recognition, data storage, and labeling, were sequentially interlinked. Consequently, the product of all these individual steps is designated as the comprehensive performance of the system, as indicated in Equation 5.2.

$$\text{Accuracy} = \frac{\text{Preprocessing} \times \text{Segmentation} \times \text{Data Augmentation} \times \text{Recognition} \times \text{Data Storage} \times \text{Labelling}}{100}$$

Equation 5.8

As previously mentioned, the system underwent a division into two distinct parts for isolated evaluations. Subsequently, these individual performances were amalgamated to determine the overall system performance, under the assumption that the steps of pre-processing, data augmentation, data storage, and labeling did not exert influence on the performance of the system. These steps were consequently excluded from the equation.

Moreover, the segmentation and recognition steps exhibited individual performances of 98.19% and 97.58%, respectively. Thus, the overall performance of the system, expressed as a percentage, was derived using Equation 5.2. In conclusion, the comprehensive performance of the system amounted to 95.81%.

$$\text{Accuracy} = \frac{98.19 \times 97.58}{100}$$

$$\text{Accuracy} = 95.81\%$$

5.5 Statistical Distribution

The Brahmi script comprises 368 distinct categories, including 33 consonants, ten vowels, and the remainder consisting of compound and intricate compound characters (Siromoney et al., 1983). However, due to lack of resources, the proposed dataset consists only 170 classes (groups) of characters with 27 varieties of consonants, 4 types of vowels, and compound characters making up the rest (139), details of the vowel, consonants, and

compound characters along with labelling are mentioned Appendix. The developed dataset has 167 and 55 text sample for training and testing, respectively. These all charatres and compound characters were isolated from 167 text were used to composed of 30,600 images designated for training and 10,200 images designated as testing. Characters from 55 samples of Brahmi text were not segmented as that will be used to test the performance of the segmentation algorithms in the future.

In this study, the resizing is done during the recognition stage. In addition, this study does not focus on the resolution of images of Brahmi dataset. However, all of the images should be visible to the naked eyes, hence, is checked during the data collection step.

5.6 Summary

The primary objective of this study was to assess the performance of the segmentation and recognition model by proposing a novel approach to enhance system accuracy and mitigate the limitations of prior research. In this regard, a combined approach using Object Detection and K-nearest neighbors (KNN) was introduced for the isolation of characters and compound characters from Brahmi text. Object detection was applied to identify all characters, including individual dots (.), which were subsequently linked to the nearest left character using KNN. This method not only improved segmentation performance but also addressed constraints observed in previous studies. Remarkable results were achieved in the segmentation approach, with object detection achieving an accuracy rate of 99.81%, while the KNN approach reached 97.32%. Overall, the average accuracy for character segmentation from the text amounted to 98.19%.

The output from the segmentation approach resulted in 40 samples per class (170 classes), distributed in a 75:25 ratio for training and testing, translating to 30 training samples and 10 testing samples per class. Moreover, data augmentation techniques were applied to

these samples to augment the quality and quantity of the dataset. These augmentation methods encompassed adjustments, Gaussian blur, deformations, the introduction of noise (salt and pepper noise, Gaussian noise), and translations. Consequently, this process generated 150 samples, equating to 50 samples per class, for the training dataset. Additionally, 50 samples, with 10 samples per category, were produced for the testing dataset using the same augmentation techniques. Following augmentation, the dataset expanded to encompass 180 samples per class for the training dataset and 60 samples per class for the testing dataset.

Lastly, a comparison between the proposed and existing systems revealed that the proposed system achieved an accuracy of 97.58%, while the existing system achieved 80.20% and 90.24%. Consequently, the introduced Brahmi text recognition system, coupled with the TPEBR process, exhibited the highest accuracy in Brahmi text recognition compared to existing methods and previous studies in this field. In conclusion, the comprehensive performance of the system amounted to 95.81%.

CHAPTER 6

CONCLUSION AND RECOMMENDATION

Brahmi script is an ancient script. Various content has been written using this script, such as the rules of the Kingdom of King Ashoka, the religious books on Buddhism, Jainism, and so on. Various texts in Brahmi script have already been converted into images via the scanning process. However, machines are unable to read the text from these images. Limited research has been dedicated to Brahmi text recognition, and the existing studies primarily relied on local datasets that are not accessible to the public. Consequently, this lack of data transparency poses challenges for researchers aiming to conduct further investigations in this domain.

This study primarily revolves around the development of a Brahmi dataset, aiming to address the scarcity of publicly available datasets and enhance transparency and research opportunities in Brahmi text recognition. The process begins with data collection from various online sources, resulting in 217 samples of text and words, and 801 isolated characters and compound characters. However, these samples lack uniformity in text and word sizes, necessitating further standardization.

The subsequent step focuses on character isolation from words and text, employing a proposed segmentation approach, a crucial precursor to system training. To augment the quality and quantity of the dataset, data augmentation techniques, including adjustments, deformations, blurring, translations, and noise introduction, are applied.

Following this, the collected samples undergo recognition through various deep learning architectures. Since each architecture requires specific input sizes, resizing is conducted to match these requirements before recognition. The recognized characters are

then automatically organized into folders, utilizing Unicode for folder labeling, where each unique Unicode code corresponds to a specific Brahmi character.

Moreover, this study introduces a combined approach for character and compound character segmentation using Object Detection and K-nearest neighbors (KNN). Object detection identifies characters, including dots (.), which are subsequently linked to their nearest left character using KNN. This approach significantly improves segmentation, achieving an impressive average accuracy of 98.19%.

The segmentation approach generates 40 samples per class (170 classes), divided into a 75:25 training-testing ratio (30 and 10 samples for training and testing, respectively). Data augmentation further increases the size and quality of the dataset, resulting in 180 training samples per class and 60 testing samples per class.

Among various deep learning architectures, SqueezeNet stands out as the most effective, with a minimal loss of 0.237 and an outstanding accuracy of 97.58%. It excels in precision, recall, and F1-score, demonstrating robust performance. In contrast, ResNeXt Small performs relatively poorly with higher loss and lower accuracy.

The study proceeds to compare the proposed process with previous studies. Comparing the proposed system to the existing system, the proposed system achieves 97.58% accuracy, while the existing system records 80.20% and 90.24%.

In conclusion, the proposed process and introduced Brahmi text recognition system exhibit superior accuracy compared to existing methods and prior Brahmi text recognition studies.

6.1 Contributions

The main contribution of this research to tackling the challenge of Brahmi text recognition is in the development of the unique multiple-network process, which can be

summarised with respect to the objectives listed below (also mentioned in Section 1.4, page no 27):

Objective 1: To establish a dataset incorporating automated processes encompassing labeling and storage.

The methodology adopted in this study for establishing a new dataset dedicated to Brahmi text recognition has significantly improved the overall performance in Brahmi text recognition. The initial dataset creation process began with sample collection from diverse online resources. Multiple online platforms were referenced to acquire these samples, with particular attention given to standardizing the text and word sizes.

Subsequent to this collection phase, a segmentation step was implemented to isolate characters from the words and text. These segmented characters were then recognized and systematically stored within dedicated directories for subsequent analysis. Furthermore, each directory was labeled with the corresponding Brahmi characters to enhance organization and categorization.

Following these data preparation steps, well-established techniques for Brahmi character and word recognition were applied to the newly created dataset.

Objective 2: To introduce and implement an approach capable of proficiently segmenting diverse characters present within Brahmi text.

In this study, a unified approach for segmenting characters and compound characters using both Object Detection and K-nearest neighbors (KNN) is presented. Object detection identifies characters, including dots (.), which are then connected to their closest left characters using KNN. This novel approach outperformed previous methods in segmenting Brahmi characters from the text and mitigating the limitations observed in earlier studies.

Objective 3: To introduce and develop a procedure for an automatic Brahmi text recognition system.

This study introduced an innovative approach for building a Brahmi text recognition system. The proposed method relies on a two-phase approach aimed at enhancing the performance of the Brahmi text recognition system. Notably, the SqueezeNet deep learning architecture was harnessed in the development of this approach, resulting in the attainment of the highest recognition rate when compared to prior research efforts.

The central contribution of this thesis lies in the development of an automated system for the creation of a dataset. The primary objectives were to establish a system, devise an innovative method for effectively segmenting Brahmi characters and compound characters from text, and enhance the performance of Brahmi text recognition systems.

The segmentation approach introduced is a unified method for the segmentation of both characters and compound characters. It leverages both Object Detection and K-nearest neighbors (KNN) to achieve this. Object detection identifies characters, including dots (.), and associates them with their nearest left characters using KNN.

Moreover, the recognition process in this methodology adopts a two-phase approach. The first phase is geared towards recognizing global features, while the second phase is dedicated to the recognition of local features. In both phases, the SqueezeNet deep learning architecture is employed.

6.2 Limitations

While this study successfully attained an accuracy of 98.19% through the segmentation approach and 97.58% in the Brahmi word recognition system, it encountered certain limitations during the course of system development, as detailed below:

6.2.1 Dataset

The dataset for this study was collected from a variety of terms and texts. However, samples of only 170 class were collected. Complex compound characters is not a part of this dataset also. Complex compound characters are when more than one vowel follows a consonant and become a new character, or two characters combines to each other and become a new character.

6.2.2 Segmentation

The segmentation approach effectively isolated Brahmi characters and compound characters from the text, demonstrating strong performance. Nevertheless, it is important to note that this approach was not evaluated on complex compound characters due to a scarcity of samples. Thus, the performance may vary when applied to complex compound characters.

6.2.3 Brahmi Text Recognition System

The Brahmi text recognition system presented in this study was unable to recognise all the classes of vowels, consonants, and compound characters of Brahmi words because samples for all of these classes could not be collected. Furthermore, this system was not designed to recognise complex compound characters.

6.3 Future Works

The limitations of the system could be addressed in the future research. These areas of improvements are presented below.

6.3.1 Dataset

The dataset did not include all vowel, consonant, compound character, and complicated compound character classes. Therefore, samples of these missing classes might be collected in order to overcome this limitation.

6.3.2 Segmentation

As there is a limited availability of samples for complex compound characters, the segmentation approach could not undergo testing with various types of these characters within the Brahmi script. Conducting tests on complex compound characters would enhance the reliability of the system.

6.3.3 Brahmi Text Recognition System

Due to the limited sample availability, the Brahmi text recognition algorithm could not identify all the distinct character classes of Brahmi words. Hence, there is a room for future improvement to enable recognition of all Brahmi character classes. Additionally, the current approach does not specifically address complex compound characters, but this limitation could be addressed in future developments.

REFERENCES

- Abbass, G. Y., & Marhoon, A. F. (2022). Car license plate segmentation and recognition system based on deep learning. *Bulletin of Electrical Engineering Informatics*, 11(4), 1983-1989. <https://doi.org/10.11591/eei.v11i4.3434>
- Abir, A. S. M., Rahman, S., Ellin, S., Farzana, M., Manik, M. H., & Rahman, C. R. (2020). Confronting the constraints for optical character segmentation from printed Bangla text image. Proceedings of the 7th International Conference on Networking, Systems and Security, Dhaka Bangladesh.
- Acharya, E. R. (2018). Evidences of hierarchy of brahmi numeral system. *Journal of the Institute of Engineering*, 14(1), 136-142. <https://doi.org/10.3126/jie.v14i1.20077>
- Adak, C., Chaudhuri, B. B., & Blumenstein, M. (2016). Offline cursive Bengali word recognition using CNNs with a recurrent model. 15th International Conference on Frontiers in Handwriting Recognition, Shenzhen, China.
- Ahmed, T., Raihan, M. N., Kushol, R., & Salekin, M. S. (2019). A complete Bangla optical character recognition system: An effective approach. 2019 22nd International Conference on Computer and Information Technology (ICCIT), Dhaka, Bangladesh.
- Ajmire, P., Dharaskar, R., & Thakare, V. (2015). Handwritten Devanagari (Marathi) compound character recognition using seventh central moment. *International Journal of Innovative Research in Computer and Communication Engineering*, 3(6). <https://doi.org/10.15680/ijircce.2015.0306078>
- Al Nasim, M. A., Ferdous, R. E., Pantho, M. A. H., & Chowdhury, A. I. (2020). A comparative analysis on Bangla handwritten digit recognition with data augmentation and non-augmentation process. 2020 International Congress on

Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey.

Allan, K. (2015). *The Routledge Handbook of Linguistics*. Taylor & Francis.
<https://books.google.com.my/books?id=TPwsCgAAQBAJ>

Ayyoob, M., & Ilyas, P. M. (2022). Efficient binarization of ancient handwritten vattezhuthu documents. 2022 3rd International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), Ghaziabad, India.

Bag, S., Harit, G., & Bhowmick, P. (2014). Recognition of Bangla compound characters using structural decomposition. *Pattern Recognition*, 47(3), 1187-1201.
<https://doi.org/10.1016/j.patcog.2013.08.026>

Basu, S., Chaudhuri, C., Kundu, M., Nasipuri, M., & Basu, D. K. (2004). A two-pass approach to pattern classification. International Conference on Neural Information Processing (ICONIP) 2004, Calcutta, India.

Bhatta, S., Srivastava, H., Das, S. K., & Singh, P. (2021). License plate detection for smart parking management. International Conference on Emerging Trends and Advances in Electrical Engineering and Renewable Energy, Bhubaneswar, India.

Bhowmik, S., Malakar, S., Sarkar, R., Basu, S., Kundu, M., & Nasipuri, M. (2019). Off-line Bangla handwritten word recognition: a holistic approach. *Neural Computing and Applications*, 31(10), 5783-5798. <https://doi.org/10.1007/s00521-018-3389-1>

Bhowmik, S., Malakar, S., Sarkar, R., & Nasipuri, M. (2014a). Handwritten Bangla word recognition using elliptical features. International Conference on Computational Intelligence and Communication Networks, Bhopal, India.

- Bhowmik, S., Roushan, M. G., Sarkar, R., Nasipuri, M., Polley, S., & Malakar, S. (2014b). Handwritten Bangla word recognition using HOG descriptor. Fourth International Conference of Emerging Applications of Information Technology, Kolkata, India.
- Bhowmik, T. K., Roy, U., & Parui, S. K. (2012). Lexicon reduction technique for Bangla handwritten word recognition. 10th International Workshop on Document Analysis Systems, Gold Cost, QLD, Australia.
- Bilal, M., Maqsood, M., Yasmin, S., Hasan, N. U., & Rho, S. (2022). A transfer learning-based efficient spatiotemporal human action recognition framework for long and overlapping action classes. *The Journal of Supercomputing*, 78(2), 2873-2908. <https://doi.org/10.1007/s11227-021-03957-4>
- Bosworth, B. (2003). The genitics of civilization: an empirical classification of civilizations based on writing systems. *Comparative Civilizations Review*, 49(49), 3.
- Brindha, S., & Bhuvaneswari, S. (2021). Repossession and recognition system: transliteration of antique Tamil Brahmi typescript. *Current Science*, 120(4), 654-665.
- Budhi, R. K., Hidayat, D. T., & Pangapul, J. P. (2018). Development of English-Indonesia idiom mobile dictionary using optical character recognition. *Jurnal Transformatika*, 16(1), 14-21. <https://doi.org/10.26623/transformatika.v16i1.869>
- Chaudhuri, A., Mandaviya, K., Badelia, P., & Ghosh, S. K. (2017). Optical character recognition systems. In *Optical Character Recognition Systems for Different Languages with Soft Computing* (pp. 9-41). Springer.
- Chaudhuri, B. (2006). A complete handwritten numeral database of Bangla - A major Indic script. Tenth International Workshop on Frontiers in Handwriting Recognition, La Baule, France.

- Dargan, S., Kumar, M., Ayyagari, M. R., & Kumar, G. (2020). A survey of deep learning and its applications: a new paradigm to machine learning. *Archives of Computational Methods in Engineering*, 27(4), 1071-1092. <https://doi.org/10.1007/s11831-019-09344-w>
- Das, M. S., Rao, K. R. M., & Balaji, P. (2019). Neural-based hit-count feature extraction method for telugu script optical character recognition. In *Innovations in Electronics and Communication Engineering* (pp. 479-486). Springer.
- Das, N., Acharya, K., Sarkar, R., Basu, S., Kundu, M., & Nasipuri, M. (2014). A benchmark image database of isolated Bangla handwritten compound characters. *International Journal on Document Analysis and Recognition*, 17(4), 413-431. <https://doi.org/10.1007/s10032-014-0222-y>
- Das, N., Basu, S., Sarkar, R., Kundu, M., Nasipuri, M., & Basu, D. k. (2009). An improved feature descriptor for recognition of handwritten bangla alphabet. International Conference on Signal and Image Processing, Excel India Publishers, Mysore, India.
- Das, N., Das, B., Sarkar, R., Basu, S., Kundu, M., & Nasipuri, M. (2010). Handwritten Bangla basic and compound character recognition using MLP and SVM classifier. *Journal of Computing*, 2(2), 109-115.
- Das, N., Sarkar, R., Basu, S., Saha, P. K., Kundu, M., & Nasipuri, M. (2015). Handwritten Bangla character recognition using a soft computing paradigm embedded in two pass approach. *Pattern Recognition*, 48(6), 2054-2071. <https://doi.org/10.1016/j.patcog.2014.12.011>
- Devi, P. D., & Sathiyapriya, V. (2021). Brahmi script recognition system using deep learning techniques. 2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India.

- Dongre, V. J., & Mankar, V. H. (2012). Development of comprehensive Devanagari numeral and character database for offline handwritten character recognition. *Applied Computational Intelligence and Soft Computing*, 2012, 1-6. <https://doi.org/10.1155/2012/871834>
- Edem, S. (2021). Handwritten Telugu Script Recognition: A Deep Learning Approach. *GIS Science Journal*, 8(11), 1466-1474.
- Ferdous, J., Karmaker, S., Rabby, A. S. A., & Hossain, S. A. (2021). Matrivaasha: A multipurpose comprehensive database for bangla handwritten compound characters. *Emerging Technologies in Data Mining and Information Security*, Kolkata, India.
- Garg, N. K., Kaur, L., & Jindal, M. (2013). Recognition of offline handwritten Hindi text using SVM. *International Journal of Image Processing*, 7(4), 395-401.
- Gautam, N., & Chai, S. S. (2017). Optical character recognition for Brahmi script using geometric method. *Journal of Telecommunication, Electronic and Computer Engineering*, 9(3-11), 131-136.
- Gautam, N., Sharma, R. S., & Hazrati, G. (2016). Handwriting recognition of Brahmi script (an artefact): Base of PALI language. In *Proceedings of First International Conference on Information and Communication Technology for Intelligent Systems* (Vol. 2, pp. 519-527). Springer, Cham.
- Gayathri Devi, S., Vairavasundaram, S., Teekaraman, Y., Kuppusamy, R., & Radhakrishnan, A. (2022). A deep learning approach for recognizing the cursive Tamil characters in palm leaf manuscripts. *Computational Intelligence Neuroscience*, 2022, 1-15. <https://doi.org/10.1155/2022/3432330>

- Ghaleb, H., Nagabhushan, P., & Pal, U. (2017). Segmentation of offline handwritten Arabic text. 2017 1st International Workshop on Arabic Script Analysis and Recognition (ASAR), Nancy, France.
- Ghosh, D., Dube, T., & Shivaprasad, A. (2010). Script recognition - a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12), 2142-2161.
<https://doi.org/10.1109/TPAMI.2010.30>
- Golait, M. S. S., Malik, L., & Thomas, A. (2017). Handwritten marathi compound character recognition using combined discrete wavelet transform and edge map features. *International Journal of Computer Science and Information Security*, 15(10), 138-146.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA.
- Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., & Vasudevan, V. (2019). Searching for mobilenetv3. Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea (South).
- Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA.
- Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size. *arXiv preprint arXiv:1602.07360*.
- Igunma, J. (2018). *Buddhism Illuminated: Manuscript Art from South-East Asia*. University of Washington Press.

- Jindal, K., & Kumar, R. (2018). A new method for segmentation of pre-detected Devanagari words from the scene images: Pihu method. *Computers & Electrical Engineering*, 70, 754-763. <https://doi.org/10.1016/j.compeleceng.2017.12.017>
- Kaur, I., & Singh, K. J. (2016). Printed Text Recognition System for Multi-Script Image.
- Keay, J. (2008). *India: A History*. Paw Prints. <https://books.google.com.my/books?id=rBlWPgAACAAJ>
- Koerich, A. L., Sabourin, R., & Suen, C. Y. (2003). Large vocabulary off-line handwriting recognition: A survey. *Pattern Analysis & Applications*, 6(2), 97-121. <https://doi.org/10.1007/s10044-002-0169-3>
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), 84-90. <https://doi.org/10.1145/3065386>
- Kumar, J. R., Sujatha, B., & Leelavathi, N. (2021). Automatic vehicle number plate recognition system using machine learning. IOP Conference Series: Materials Science and Engineering,
- Kumar, R., Kumar, A., & Ahmed, P. (2013). A benchmark dataset for Devnagari document recognition research. 6th International Conference on Visualization, Imaging and Simulation, Lemesos: Cyprus.
- Kushwaha, A. K., & Singh, A. P. (2020). *Preserving complex digital objects in the GLAM community through Digital Humanities: A study on Ancient Indian scripts* International Federation of Library Associations and Institutions (IFLA), World Library and Information Congress (WLIC), Dublin, Ireland.

- Madhvanath, S., & Govindaraju, V. (2001). The role of holistic paradigms in handwritten word recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2), 149-164. <https://doi.org/10.1109/34.908966>
- Madhvanath, S., Kleinberg, E., & Govindaraju, V. (1999). Holistic verification of handwritten phrases. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(12), 1344-1356. <https://doi.org/10.1109/34.817412>
- Maharana, K., Mondal, S., & Nemade, B. (2022). A review: Data pre-processing and data augmentation techniques. *Global Transitions Proceedings*, 3(1), 91-99.
- Mahender, C. N., & Kale, K. (2011). Structured based feature extraction of handwritten Marathi word. *International Journal of Computer Applications*, 16(6), 42-47. <https://doi.org/10.5120/2013-2718>
- Maheswari, P. U., Aswathy, A., Ezhilarasi, S., & Priya, M. R. (2022). Recognition of vowels, consonants and compound character sequences from ancient tamil stone inscriptions using deep neural networks. 2022 IEEE International Power and Renewable Energy Conference (IPRECON), Kollam, India.
- Majid, N., & Smith, E. H. B. (2019). Segmentation-free Bangla offline handwriting recognition using sequential detection of characters and diacritics with a faster R-CNN. 2019 International Conference on Document Analysis and Recognition (ICDAR), Sydney, NSW, Australia.
- Memon, J., Sami, M., Khan, R. A., & Uddin, M. (2020). Handwritten optical character recognition (OCR): A comprehensive systematic literature review (SLR). *IEEE Access*, 8, 142642-142668. <https://doi.org/10.1109/ACCESS.2020.3012542>
- Michalak, H., & Okarma, K. (2019). Fast binarization of unevenly illuminated document images based on background estimation for optical character recognition purposes.

- Mumuni, A., & Mumuni, F. (2022). Data augmentation: A comprehensive survey of modern approaches. *Array*, 16, 100258. <https://doi.org/10.1016/j.array.2022.100258>
- Nagane, A. S., & Mali, S. (2020). Segmentation of characters from degraded brahmi script images. *Applied Computer Vision and Image Processing*, Nanded, India.
- Nagane, A. S., & Mali, S. M. (2021). Identification and segmentation of touching brahmi characters from degraded digital estampage images using ensemble classifier. *Indian Journal of Computer Science and Engineering (IJCSE)*, 12(6), 1722-1733. <https://doi.org/10.21817/indjcse/2021/v12i6/211206110>
- Nagendra Kumar, S., Revathi, B., Vidya, P., & Neha, S. C. (2022). Optical character recognition of telugu text using inception model. *Proceedings of Third International Conference on Intelligent Computing, Information and Control Systems: ICICCS 2021*, Secunderabad, India.
- Nandam, S. R., & Negi, A. (2022). A versatile scene text recognition approach for telugu. *Available at SSRN 4216201*.
- Narang, S. R., Jindal, M., & Kumar, M. (2020). Line segmentation of Devanagari ancient manuscripts. *Proceedings of the National Academy of Sciences, India section A: physical sciences*, 90, 717-724. <https://doi.org/10.1007/s40010-019-00627-2>
- Narang, S. R., Jindal, M. K., & Kumar, M. (2019). Drop flow method: An iterative algorithm for complete segmentation of Devanagari ancient manuscripts. *Multimedia Tools and Applications*, 78, 1-26. <https://doi.org/10.1007/s11042-019-7620-6>
- Obaidullah, S. M., Halder, C., Das, N., & Roy, K. (2016). PWDB_13: A corpus of word-level printed document images from thirteen official indic scripts. *Proceedings of the*

- 4th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA) 2015, Durgapur, India.
- Obaidullah, S. M., Halder, C., Santosh, K., Das, N., & Roy, K. (2018). PHDIndic_11: page-level handwritten document image dataset of 11 official Indic scripts for script identification. *Multimedia Tools and Applications*, 77(2), 1643-1678. <https://doi.org/10.1007/s11042-017-4373-y>
- Pal, U., & Chaudhuri, B. (2004). Indian script character recognition: A survey. *Pattern Recognition*, 37(9), 1887-1899. <https://doi.org/10.1016/j.patcog.2004.02.003>
- Phiphatphaisit, S., & Surinta, O. (2020). Food image classification with improved MobileNet architecture and data augmentation. Proceedings of the 3rd International Conference on Information Science and Systems, Cambridge, United Kingdom.
- Plötz, T., & Fink, G. A. (2009). Markov models for offline handwriting recognition: A survey. *International Journal on Document Analysis and Recognition*, 12(4), 269-298. <https://doi.org/10.1007/s10032-009-0098-4>
- Pramanik, R., & Bag, S. (2018). Shape decomposition-based handwritten compound character recognition for Bangla OCR. *Journal of Visual Communication and Image Representation*, 50, 123-134. <https://doi.org/10.1016/j.jvcir.2017.11.016>
- Pramanik, R., & Bag, S. (2020). Segmentation-based recognition system for handwritten Bangla and Devanagari words using conventional classification and transfer learning. *IET Image Processing*, 14(5), 959-972. <https://doi.org/10.1049/iet-ipr.2019.0208>
- Preethi, P., & Mamatha, H. (2023). Intelligent Recognition of Ancient Brahmi Characters using transfer learning. 2023 Third International Conference on Advances in

Electrical, Computing, Communication and Sustainable Technologies (ICAECT),
Bhilai, India.

- Preethi, P., & Mamatha, H. R. (2023). Region-based convolutional neural network for segmenting text in epigraphical images. *Artificial Intelligence and Applications*, 1(2), 119-127. <https://doi.org/10.47852/bonviewAIA2202293>
- Qazi, E. U. H., Zia, T., & Almorjan, A. (2022). Deep learning-based digital image forgery detection system. *Applied Sciences*, 12(6), 2851. <https://doi.org/10.3390/app12062851>
- Rajan, V. (2010). *Virtual vinodh*. <http://www.virtualvinodh.com/wp/>
- Rajyagor, B., & Rakholia, R. (2021). Tri-level handwritten text segmentation techniques for Gujarati language. *Indian Journal of Science and Technology*, 14(7), 618-627. <https://doi.org/10.17485/IJST/v14i7.2146>
- Ramachandrula, S., Jain, S., & Ravishankar, H. (2012). Offline handwritten word recognition in Hindi. Proceeding of the Workshop on Document Analysis and Recognition, Mumbai, India.
- Roy, A., & Mandal, M. (2016). *Brahmi: Rediscovering the Lost Script*. <https://books.google.com.my/books?id=2c0ynQAACAAJ>
- Roy, P. P., Bhunia, A. K., Das, A., Dey, P., & Pal, U. (2016). HMM-based Indic handwritten word recognition using zone segmentation. *Pattern Recognition*, 60, 1057-1075. <https://doi.org/10.1016/j.patcog.2016.04.012>
- Roy, P. P., Dey, P., Roy, S., Pal, U., & Kimura, F. (2014). A novel approach of Bangla handwritten text recognition using HMM. 14th International Conference on Frontiers in Handwriting Recognition, Heraklion, Greece.

- Roy, S., Das, N., Kundu, M., & Nasipuri, M. (2017). Handwritten isolated Bangla compound character recognition: A new benchmark using a novel deep learning approach. *Pattern Recognition Letters*, 90, 15-21. <https://doi.org/10.1016/j.patrec.2017.03.004>
- Sahare, P., & Dhok, S. B. (2019). Robust character segmentation and recognition schemes for multilingual Indian document images. *IETE Technical Review*, 36(2), 209-222.
- Salomon, R. (1998). *Indian epigraphy: A guide to the study of inscriptions in Sanskrit, Prakrit, and the other Indo-Aryan languages*. Oxford University Press.
- Sánchez, J. A., & Pal, U. (2016). Handwritten text recognition for Bengali. 15th International Conference on Frontiers in Handwriting Recognition, Shenzhen, China.
- Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA.
- Sarkar, R., Das, N., Basu, S., Kundu, M., Nasipuri, M., & Basu, D. K. (2012). CMATERdb1: A database of unconstrained handwritten Bangla and Bangla-English mixed script document image. *International Journal on Document Analysis and Recognition*, 15(1), 71-83. <https://doi.org/10.1007/s10032-011-0148-6>
- Sarkhel, R., Saha, A. K., & Das, N. (2015). An enhanced harmony search method for Bangla handwritten character recognition using region sampling. 2nd International Conference on Recent Trends in Information Systems, Kolkata, India.
- Sathik, M. M., & Ratheash, R. S. (2021). Text line segmentation in tamil language palm leaf manuscripts-a novel approach. *Journal of Tianjin University Science and Technology*, 54(4), 297-304. <https://doi.org/10.17605/OSF.IO/8DWSQ>
- Satyawan, W., Pratama, M. O., Jannati, R., Muhammad, G., Fajar, B., Hamzah, H., Fikri, R., & Kristian, K. (2019). Citizen ID card detection using image processing and

- optical character recognition. *Journal of Physics: Conference Series*, 1235, 012049.
<https://doi.org/10.1088/1742-6596/1235/1/012049>
- Seland, E. H. (2014). Archaeology of trade in the western Indian Ocean, 300 BC–AD 700. *Journal of Archaeological Research*, 22(4), 367-402.
<https://doi.org/10.1007/s10814-014-9075-7>
- Sharma, R., & Mudgal, T. (2019). Primitive Feature-Based Optical Character Recognition of the Devanagari Script. Progress in Advanced Computing and Intelligent Engineering, Odisha, India.
- Shaw, B., Parui, S. K., & Shridhar, M. (2008). Offline Handwritten Devanagari Word Recognition: A holistic approach based on directional chain code feature and HMM. International conference on information technology, Bhubaneswar, India.
- Shorten, C., Khoshgoftaar, T. M., & Furht, B. (2021). Text data augmentation for deep learning. *Journal of Big Data*, 8, 1-34. <https://doi.org/10.1186/s40537-021-00492-0>
- Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- Singh, A. P., & Kushwaha, A. K. (2019). Analysis of segmentation methods for brahmi script. *DESIDOC Journal of Library & Information Technology*, 39(2), 109-116.
<https://doi.org/10.14429/djlit.39.2.13615>
- Singh, U. (2008). *A History of Ancient and Early Medieval India: From the Stone Age to the 12th Century*. Pearson Education.
- Siromoney, G., Chandrasekaran, R., & Chandrasekaran, M. (1983). Machine recognition of Brahmi script. *IEEE Transactions on Systems Man and Cybernetics*, SMC-13(4), 648-654. <https://doi.org/10.1109/TSMC.1983.6313155>

- Soumya, A., & Kumar, G. H. (2015). Recognition of historical records using gabor and zonal features. *Signal and Image Processing: An International Journal*, 6(4), 57-69. <https://doi.org/10.5121/sipij.2015.6405>
- Srivastav, A., & Sahu, N. (2016). Segmentation of Devanagari handwritten characters. *International Journal of Computer Applications*, 142(14).
- Subadivya, S., Vigneswari, J., Yaminie, M., & Diviya, M. (2020). Tamil-Brahmi script character recognition system using Deep Learning technique. *International Journal of Computer Science and Mobile Computing*, 9(6), 114-119.
- Subba, J. R. (2008). *Evaluation Of Man And The Modern Society In Sikkim*. Gyan Publishing House. <https://books.google.com.my/books?id=q67nHqe4idQC>
- Sulaiman, A., Omar, K., & Nasrudin, M. F. (2019). Degraded historical document binarization: A review on issues, challenges, techniques, and future directions. *Journal of Imaging*, 5(4), 48. <https://doi.org/10.3390/jimaging5040048>
- Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, California, USA.
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA.
- Tabassum, S., Abedin, N., Rahman, M. M., Rahman, M. M., Ahmed, M. T., Islam, R., & Ahmed, A. (2022). An online cursive handwritten medical words recognition system for busy doctors in developing countries for ensuring efficient healthcare service delivery. *Scientific Reports*, 12(1), 3601. <https://doi.org/10.1038/s41598-022-07571-z>

- Vellingiriraj, E., Balamurugan, M., & Balasubramanie, P. (2016). Text analysis and information retrieval of historical tamil ancient documents using machine translation in image zoning. *International Journal of Languages Literature and Linguistics*, 2(4), 164-168. <https://doi.org/10.18178/IJLLL.2016.2.4.88>
- Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA.
- Yang, J. (2020). Gridmask based data augmentation for bengali handwritten grapheme classification. Proceedings of the 2020 2nd International Conference on Intelligent Medicine and Image Processing, Association for Computing Machinery (ACM).
- Yao, N., Ni, F., Wang, Z., Luo, J., Sung, W. K., Luo, C., & Li, G. (2021). L2MXception: An improved Xception network for classification of peach diseases. *Plant Methods*, 17(1), 1-13.
- Yousaf, A., Khan, M. J., Khan, M. J., Javed, N., Ibrahim, H., Khurshid, K., & Khurshid, K. (2019). Size invariant handwritten character recognition using single layer feedforward backpropagation neural networks. 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), Sukkur, Pakistan.
- Yousefi, J. (2011). Image binarization using otsu thresholding algorithm. *Ontario, Canada: University of Guelph*, 1-4.

APPENDICES

Details of the vowel, consonants, and compound characters where NA means not available

Vowel	ደ	ቀ	፡	NA	NA	NA	፱	NA	NA	NA
-------	---	---	---	----	----	----	---	----	----	----

Consonant	ተ	NA	ላ	ገ	ፒ	ዐ	ቀ	ደ	ዞ	ከ	ር
	ዐ	NA	ሪ	ፒ	ላ	ዐ	ቀ	ደ	ዞ	ከ	NA
	NA	ፒ	ሪ	NA	NA	ፒ	ቀ	NA	ደ	ዞ	ከ

Compound character

	ደ	ቀ	፡	፡፡	ፒ	ፒ	፱	፱	ፒ	ደ:
ተ	ተ	ቀ	፡	፡፡	ፒ	ፒ	፱	NA	፱	ደ:
ገ	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
ላ	ላ	NA	NA	NA	NA	NA	NA	NA	NA	NA
ገ	ገ	ፒ	ፒ	ፒ	ፒ	ፒ	ፒ	NA	ፒ	ፒ
ፒ	ፒ	ፒ	ፒ	ፒ	ፒ	NA	ፒ	NA	NA	ፒ
ዐ	ዐ	ፒ	ፒ	ፒ	NA	NA	ፒ	NA	NA	ፒ
ቀ	ቀ	ፒ	ፒ	ፒ	NA	NA	ፒ	NA	NA	ፒ
ደ	ደ	ፒ	ፒ	ፒ	ፒ	NA	NA	NA	ፒ	ፒ
ዞ	ዞ	NA	NA	NA	NA	NA	NA	NA	NA	ዞ
ከ	ከ	ፒ	NA	NA	NA	NA	NA	NA	NA	ከ
ር	ር	ፒ	ፒ	ፒ	ፒ	NA	NA	NA	NA	ር
ዐ	ዐ	ፒ	ፒ	ፒ	NA	NA	NA	NA	NA	ፒ
ፒ	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
ሪ	ሪ	ፒ	ፒ	ፒ	ሪ	NA	NA	NA	NA	ሪ
ፒ	ፒ	ፒ	ፒ	ፒ	ፒ	ፒ	ፒ	NA	ፒ	ፒ
ላ	ላ	ፒ	ፒ	ፒ	ላ	NA	ፒ	NA	ፒ	ላ

Table Continued

⊙	⊙	⊙	⊙	⊙	NA	NA	NA	NA	NA	NA
⋈	⋈	⋈	⋈	⋈	⋈	⋈	⋈	NA	⋈	⋈
⋈	⋈	⋈	⋈	⋈	NA	NA	NA	NA	⋈	⋈
⋈	⋈	⋈	⋈	⋈	⋈	⋈	⋈	NA	⋈	⋈
⋈	⋈	⋈	⋈	⋈	⋈	NA	⋈	NA	NA	⋈
⋈	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
⋈	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
⋈	⋈	NA	NA	NA	NA	NA	NA	NA	NA	⋈
⋈	⋈	⋈	⋈	NA	⋈	NA	⋈	NA	⋈	⋈
⋈	⋈	⋈	⋈	⋈	NA	NA	⋈	NA	⋈	NA
⋈	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
⋈	⋈	⋈	⋈	⋈	⋈	⋈	⋈	NA	⋈	⋈
⋈	⋈	⋈	⋈	⋈	⋈	NA	NA	NA	NA	⋈
⋈	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
⋈	⋈	NA	NA	NA	NA	NA	NA	NA	NA	NA
⋈	⋈	⋈	⋈	⋈	⋈	⋈	⋈	NA	⋈	⋈
⋈	⋈	⋈	⋈	NA	NA	NA	NA	NA	NA	⋈
Total	27	23	22	20	17	8	14	0	12	23