DESIGN OF AN OFFLINE HANDWRITING RECOGNITION SYSTEM TESTED ON THE BANGLA AND KOREAN SCRIPTS

by
Nishatul Majid

A dissertation

submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy in Electrical and Computer Engineering

Boise State University

August 2020

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the dissertation submitted by

Nishatul Majid

Dissertation Title: Design of an Offline Handwriting Recognition System Tested on the Bangla and Korean Scripts

Date of Final Oral Examination: 27 April 2020

The following individuals read and discussed the dissertation submitted by student Nishatul Majid, and they evaluated the presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Dr. Elisa H. Barney Smith, PhD	Chair, Supervisory Committee
Dr. Jennifer Smith, PhD	Member, Supervisory Committee
Dr. Nadar Rafla, PhD	Member, Supervisory Committee
Dr. Laurence Likforman-Sulem, PhD	Member (External), Supervisory Committee

The final reading approval of the thesis was granted by Elisa H. Barney, Ph.D., Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

ABSTRACT

This dissertation presents a flexible and robust offline handwriting recognition system which is tested on the Bangla and Korean scripts. Offline handwriting recognition is one of the most challenging and yet to be solved problems in machine learning. While a few popular scripts (like Latin) have received a lot of attention, many other widely used scripts (like Bangla) have seen very little progress. Features such as connectedness and vowels structured as diacritics make it a challenging script to recognize. A simple and robust design for offline recognition is presented which not only works reliably, but also can be used for almost any alphabetic writing system. The framework has been rigorously tested for Bangla and demonstrated how it can be transformed to apply to other scripts through experiments on the Korean script whose two-dimensional arrangement of characters makes it a challenge to recognize.

The base of this design is a character spotting network which detects the location of different script elements (such as characters, diacritics) from an unsegmented word image. A transcript is formed from the detected classes based on their corresponding location information. This is the first reported lexicon-free offline recognition system for Bangla and achieves a Character Recognition Accuracy (CRA) of 94.8%. This is also one of the most flexible architectures ever presented. Recognition of Korean was achieved with a 91.2% CRA. Also, a powerful technique of autonomous tagging was developed which can drastically reduce

the effort of preparing a dataset for any script. The combination of the character spotting method and the autonomous tagging brings the entire offline recognition problem very close to a singular solution.

Additionally, a database named the Boise State Bangla Handwriting Dataset was developed. This is one of the richest offline datasets currently available for Bangla and this has been made publicly accessible to accelerate the research progress. Many other tools were developed and experiments were conducted to more rigorously validate this framework by evaluating the method against external datasets (CMATERdb 1.1.1, Indic Word Dataset and REID2019: Early Indian Printed Documents). Offline handwriting recognition is an extremely promising technology and the outcome of this research moves the field significantly ahead.

ACKNOWLEDGMENTS

I'd like to express my sincerest gratitude to my PhD advisor, Dr. Elisa H. Barney Smith for all her support and encouragement during my doctoral program. I'd also like to thank my committee members Dr. Nader Rafla and Dr. Jennifer Anne Smith as well as the department of Electrical and Computer Engineering, Boise State University for all their cooperation and guidance. I'd like to specially thank all the volunteers that took selfless part in my Boise State Handwriting Dataset project. Also like to thank the Center for Microprocessor Application for Training Education and Research (CMATER) research laboratory at Jadavpur University, Kolkata, India, Indian Statistical Institute (ISI), Kolkata, India, the BanglaLekha-Isolated project funded by the ICT division, Bangladesh for their handwritten Bangla isolated basic character datasets and Pradeep Kumar for their Handwritten Bangla Word Dataset. Another special thanks to Steven Kim, Department of Computer Science, Boise State University, Boise, Idaho, USA, for his participation and contribution with the Korean handwriting recognition project.

Lastly, I'd like to acknowledge the high-performance computing support of the R2 Compute Cluster provided by Boise State University's Research Computing Department. Without this support, most of the research work completed would not be possible to achieve.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BLI – Bangla-Lekha Isolated

BLSTM/BDLSTM – Bidirectional Long Short-Term Memory

BoF – Bag of Features

BoVW – Bag of Visual Words

BoW – Bag of Words

BP – Back Propagation

CBDAR – Camera Based Document Analysis and Recognition

CER – Character Error Rate

CMATER – Center for Microprocessor Application for Training Education and Research

C-Net – Character Network

CNN – Convolutional Neural Network

CRA – Character Recognition Accuracy

DAG – Direct Acyclic Graph

DL – Deep Learning

D-Net – Diacritic Network

FC – Fully Connected

GA – Genetic Algorithm

HKS – Heat Kernel Signature

HLA – Hierarchical Learning Approach

HMM – Hidden Markov Model

HOG – Histogram Oriented Gradients

HWR – Handwriting Recognition

ICDAR – International Conference on Document Analysis and Recognition

ICFHR – International Conference on Frontiers in Handwriting Recognition

ICT – Information and Communication Technology

ILSVRC – ImageNet Large Scale Visual Recognition Competition

IoU – Intersection Over Union

IRB – Institutional Review Board

ISI – Indian Statistical Institute

JRA – Jamo Recognition Accuracy

KNN – K Nearest Neighbor

LGH – Local Gradient Histogram

LoG – Laplacian of Gaussian

LSTM – Long Short-Term Memory

mAP – Mean Average Precision

MKL – Multiple Kernel Learning

MLP – MultiLayer Perceptron

MQDF – Modified Quadratic Discriminant Function

NN – Neural Network

NSHP-HMM – Non-Symmetric Half Plane Hidden Markov Model

OCR – Optical Character Recognition

OHR – Offline Handwriting Recognition

OLCR – Online Character Recognition

OVA – One Versus All

OVO – One Versus One

PCA – Principle Component Analysis

PHOG – Pyramid Histogram of Oriented Gradient

RBF – Radial Basis Function

R-CNN – Convolutional Neural Network with Region

ReLU – Rectified Linear Unit

ResNet – Residual Neural Network

RNN – Recurrent Neural Network

RoI – Region of Interest

ROVER – Recognition Output Voting Error Reduction

RPN – Region Proposal Network

SGDM – Stochastic Gradient Descent with Momentum

SIFT – Scale-Invariant Feature Transform

SLLE – Supervised Locally Linear Embedding

SPM – Spatial Pyramid Matching

SRA – Syllable Recognition Accuracy

SVM – Support Vector Machine

USURF – Upright Speed Up Robust Features

VGG – Visual Geometry Group

WER – Word Error Rate

WRA – Word Recognition Accuracy

CHAPTER 1

INTRODUCTION

1.1 What is Offline Handwriting Recognition

Handwriting Recognition (HWR) is generally known as the capability of a computer to interpret text information from a handwritten input source, such as paper documents, photographs, touch based digital devices etc. This has always been considered to be harder than Optical Character Recognition (OCR) of machine print because of the inherent variability and randomness of writing styles. There are two major classes of HWR, one being online and the other is offline recognition. Online Character Recognition (OLCR) is when a person writes, usually with a stylus or his/her finger, on a touch sensitive device, and the computer applies a recognition process to that input using the time and stroke information of the characters. This is considered to be much simpler because of the availability of the time data and thus stroke information, as well as the lack of noise, skew, distortions etc. that an offline image always suffers from. Offline Handwriting Recognition (OHR) belongs to the class of Image Processing and Pattern Recognition, where text is recognized solely from digitally stored image data, usually from a scanner or a camera. An immediate advantage of offline recognition is it can be done at

any time after the document is written, even after years. But since it is not done in real time as someone writes, it can't be used for immediate text input. Offline Handwriting Recognition has been an interest of researchers in several fields, such as pattern recognition, artificial intelligence, computer vision etc, and the history goes back more than 30 years. It began with an automated postal code sorting task, but now with the increased demands for task automation, the importance is getting bigger and more significant.

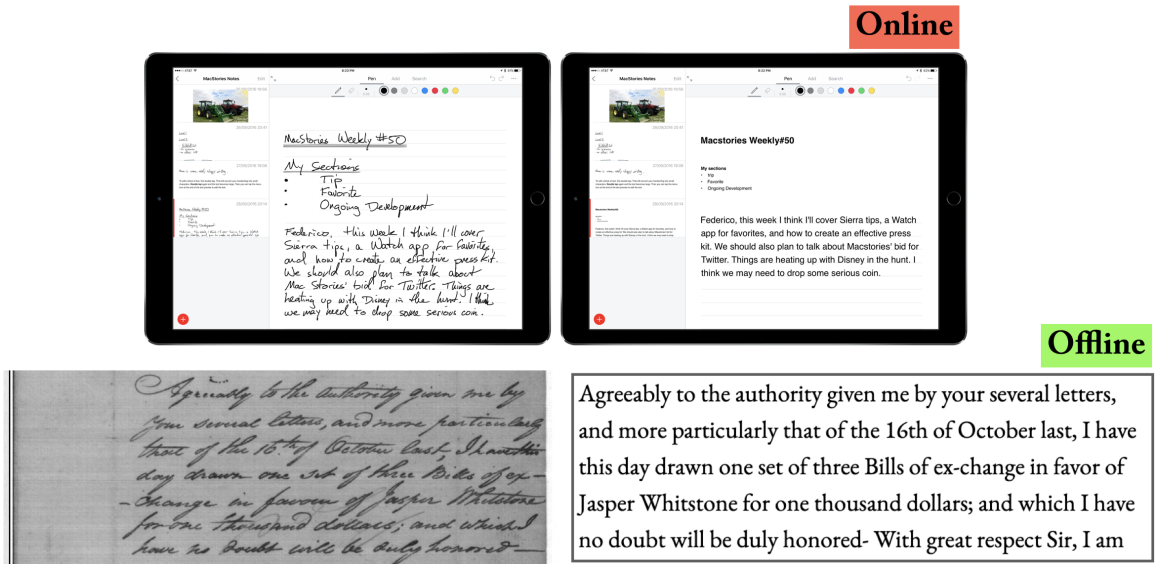
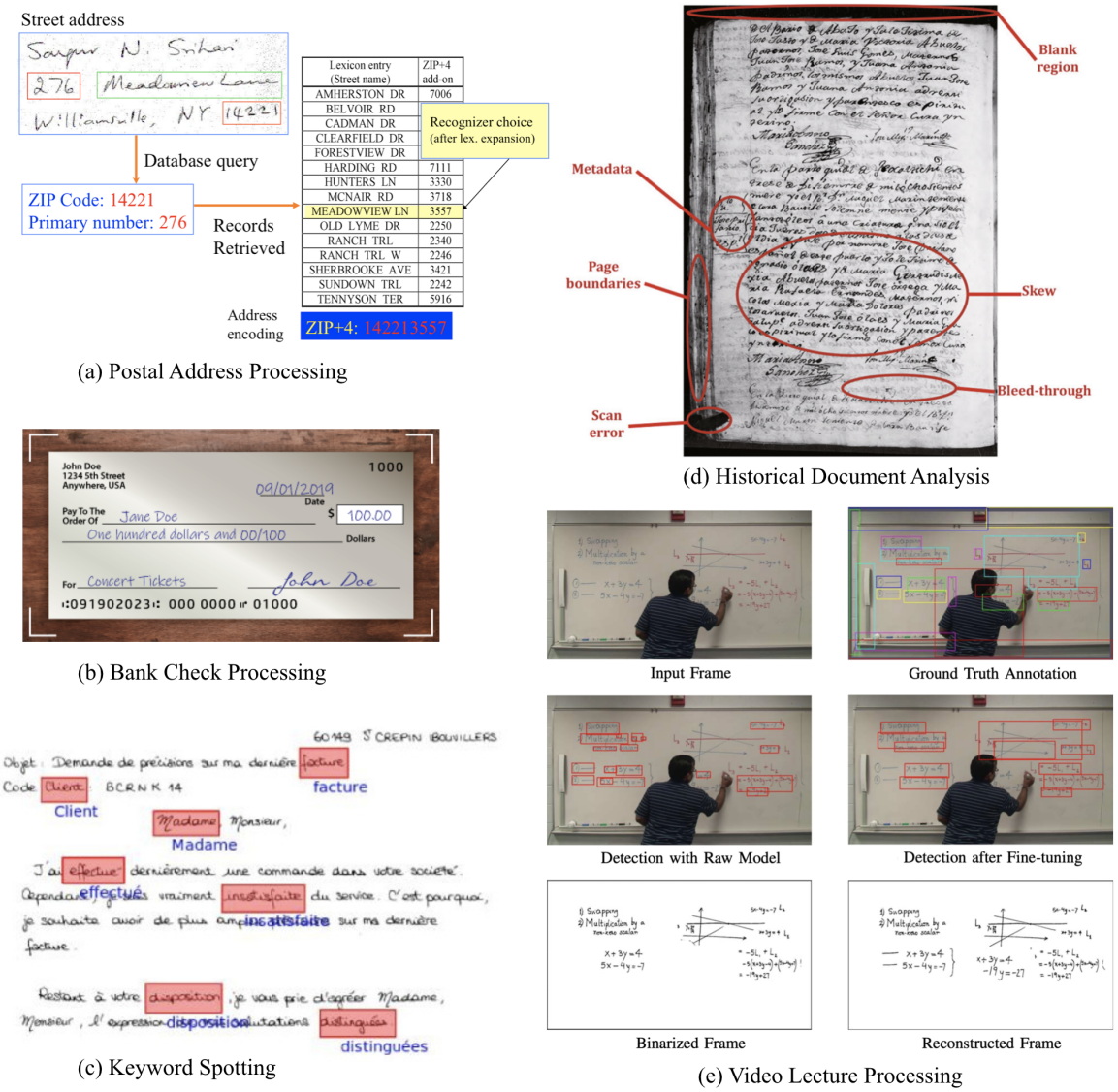


Figure 1.1: Online vs. Offline Handwriting Recognition [1].

1.2 Why Offline Handwriting Recognition

Offline handwriting recognition is still an undeveloped or at best pre-mature technology and therefore many applications and their impact in development and growth have not yet been fully explored. A few of the applications are demonstrated in Fig 1.2. One of the primary sectors that offline recognition is contributing

to today is in task automation. This includes numerous fields such as postal address verification, bank check processing, translating documents, digitization and archiving manuscripts. Several industries are adopting this technology to batch process forms and applications, and these are used in practice today. These are relatively easy because there are specific places where certain categories of data will lie, for instance the "Name" field looks only for text, whereas the "Date of Birth" field expects a formatted number. The same applies for postal addresses or bank checks. Most of the mobile client apps for top bank services provides bank check processing using cameras and they have been working very well for years now. Also, signatures can now be reliably verified from offline images. As time grows and the technology becomes faster and more accurate, it's inevitable to see a great expansion of this application area.



(a) Postal Address Processing

(b) Bank Check Processing

(c) Keyword Spotting

(d) Historical Document Analysis

(e) Video Lecture Processing

Figure 1.2: Demonstration of some applications of Offline Handwriting Recognition [2], [3], [4].

One other important area using this technology is known as keyword spotting. It finds certain keywords from document images to get an idea about the document content before further processing. This is useful in many situations, such as sorting of letters or applications, finding appropriate documents from a digital

archive, processing annotations from historical documents etc. One good example to explain this application will be if someone wants to research something, say the Second World War, and by using keyword spotting with some appropriate keywords it is possible to isolate all written manuscripts, exchanged letters and historical documents which talk about this topic.

This tool also helps us to preserve historical documents, the wisdom and saga of our ancients. Not only are these very important, they are also very tricky to deal with. Often they are very fragile and better when not touched frequently. They suffer from discoloration, partial damage and many signs of aging. Because of this special situation, many machine printed historical documents are digitized and then treated with handwriting recognition techniques for better results. In developing countries, there are often piles of documents, but not enough man power or resources to digitize or transcribe them. Many documents are destroyed everyday around the world and these could have told us the legacy of a thousand years. Offline handwriting recognition is a very powerful and potent tool that can make these historical documents much more accessible to humanity.

The advent and growth of technologies always bring new and innovative use cases that were formerly never thought of. For example, with the increased popularity of online education, handwriting recognition is now used to transcribe handwritten white board and tablet contents into machine printed and editable lecture notes. Handwriting analysis is used to monitor the progress of a kindergarten education through the handwriting development of children. There are many demographic studies that are emerging from handwriting analysis. Furthermore handwriting recognition is becoming useful in forensic studies, estimating a

person's gender, age, profession or even stress-level all from his/her handwriting samples.

Technologies and tools are developed all for one purpose: to reduce human labor, to free human-kind from the chores that can be done with low level machine intelligence. In this theme, offline handwriting recognition is vividly promising, particularly in the field of knowledge, education and research, which essentially translates into the growth and development of the entire civilization.

1.3 How Offline Recognition is Done

There are many different ways to design an offline recognizer, but they can be broadly grouped into two categories - Segmentation-based or Segmentation-free recognition. Also, the tools used can be classified into two different discussions - classical Machine Learning and modern Deep Learning based frameworks. Each of the categories and approaches has their strength and weakness. These are all briefly discussed in the following subsections.

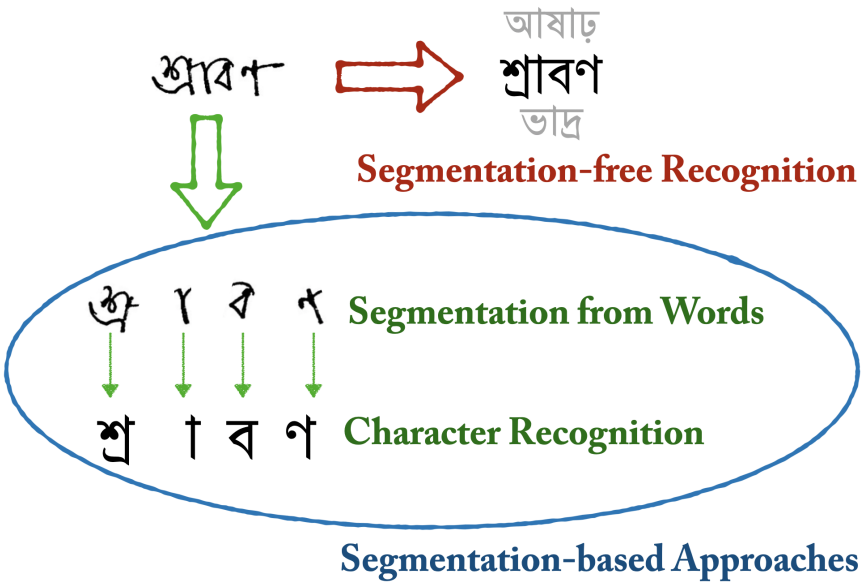


Figure 1.3: Basic ideas of Segmentation-based and Segmentation-free offline handwriting recognition. Traditional segmentation-free approaches attempt to find a match of a given word as a whole, whereas segmentation-based approaches attempt to segment the characters from a word image and find matches for each character.

1.3.1 Segmentation-Based Recognition

The idea of Segmentation-Based recognition is to first segment the document word image into individual characters and then use an isolated character recognizer to obtain a transcript of the word as shown in the bottom portion of Fig 1.3. With tools like Deep Learning, the process of classifying isolated characters is considered to be a solved problem for most scripts. The real challenge then comes from the process of character segmentation. It can be extremely difficult to achieve cleanly, which is particularly troublesome for scripts with a connected nature like Bangla, cursive Latin, Arabic and many more. It is thought that the

only sure way of isolating a character from its connected neighbors is to be able to identify or recognize the character first, which is an irony, because the whole point of segmentation is just to initiate and help the recognition process. This dilemma is famously known as Sayre's paradox [5] - which essentially implies that neither can segmentation be perfected without recognition, nor can recognition be accurate without prior segmenting for cursive or connected handwritten words.

The process of segmentation is highly script dependent. The techniques rely on some form of clever tricks applied on the formation and characteristics of a specific script. To this date, no character segmentation process has been reported that works reliably on every kind of handwriting even for a single script. But if the segmentation can be achieved somehow, the classification process that follows is relatively clean and easy.

1.3.2 Segmentation-Free Recognition

The segmentation-free approach is a very modern way of doing offline recognition. The idea is as it sounds, rather than segmenting a unit into easily recognizable chunks, this model tries to estimate the unit as a whole as shown in the top portion of Fig 1.3. It skips many complications that can arise from pre-processing stages and quite frequently it has been seen that the recognition failure is caused by poor pre-processing. Generally, segmentation-free approaches are faster and simpler, and usually produce better results. Also these approaches not only skip the character segmentation process, but also avoid many other stages like scaling, skew/slant correction, noise removal etc. Furthermore, this approach is essentially script independent, where the character segmentation requires deep integration of

each script's characteristics to work. The cost of segmentation-free processes are usually paid in terms of slower training time and higher computational complexities, which usually require powerful CPUs and GPUs for processing. Also, these approaches generally depend more on the availability and structure of datasets than the segmentation-based techniques.

1.3.3 Traditional Machine Learning

Machine learning, in the context of handwriting recognition is a process of making a system learn to recognize handwritten content from digital data. The classical or conventional way of machine learning involves a routine set of tasks. The most commonly used idea is to train a system through lots of labeled data so that it can develop the sense of class when unlabeled data comes in. This idea is known as Supervised Learning. The techniques or tools within this process can vary widely, but there are a routine set of tasks that pretty much remain constant over all approaches. These tasks are Pre-processing, Feature Extraction, Classification and Post-Processing. A schematic is shown in Fig 1.4.

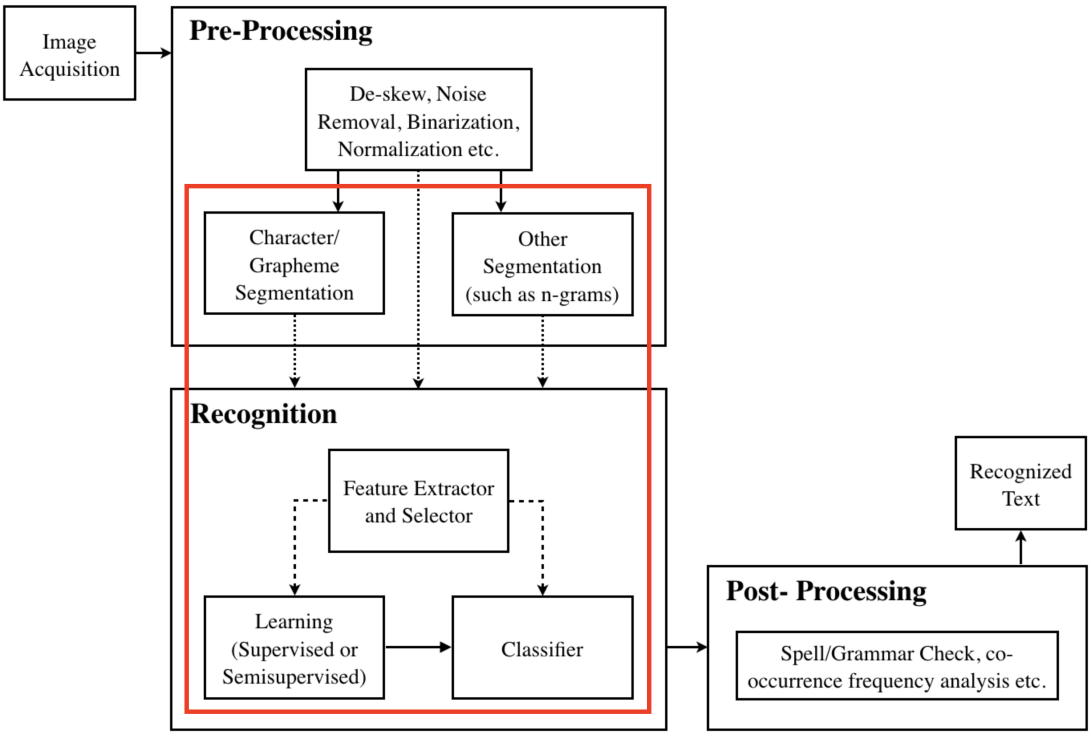


Figure 1.4: Different stages of conventional machine learning for handwriting recognition.

Preprocessing generally involves a series of operations such as de-skewing, noise removal, binarization, layout analysis, line and word detection, script recognition, normalization of aspect ratio and scale, etc. Segmentation of words into characters is also considered to be a pre-processing step. The processed images are then subjected to the Feature Extractor. Feature extraction is a process of obtaining numerical measurements of features from the handwritten images that are specific to each class. These are represented as vectors and are intended to reduce the volume of raw data for faster and better classification. Features are usually handcrafted, or chosen by a human who has expertise with the specific field. There are many techniques, a few popular ones are SIFT, SURF, gradient,

stroke directions etc. These features are appended into vectors that are then subjected to the classifier. The classifier inspects the features and estimates the class based on its prior knowledge or training. It is usually trained with labeled samples with class information which are called the training data. The method is then tested on images not previously seen which are called the test data. The classification performance is evaluated based on different metrics such as accuracy, confidence, etc. There are many developed classifiers in use such as Support Vector Machine (SVM), K-nearest neighbor, Logistic Regression, Naive Bayes, Multilayer Perceptron (MLP), etc. Often when experimenting with features or classifiers, some data from the training set is held back for performance evaluation of the system. This portion of labeled data is called the validation set. This is used to avoid overtraining on the test set which often leads to a biased score.

1.3.4 Deep Learning

The process called Deep Learning is considered to be a modern way of doing machine learning. The key idea here is to develop an end to end system, getting rid of all the stages like preprocessing, feature extraction, classification, etc. It prepares a system which is considered to be a black box, i.e. what's happening inside is not controllable via direct means. Inside the black box there are multiple layers of Artificial Neurons all connected to each other, mimicking the basic construction of a human brain. A typical structure of a Neural Network is shown in Fig 1.5, where there are multiple arrays of hidden neural layers. Usually, such a neural architecture is referred to as "deep" if there are a big number of hidden layers in the design. After constructing such a system, it is subjected to the trained or labeled

data. Once the data crosses all the layers inside the neural network, it generates an arbitrary output. Then a technique called back propagation is applied in order to tune all the parameters inside the network to a point where the network generates the desired output when the input data is applied again. This process goes on for all the training data over multiple passes, which is called the number of epochs. After several iterations, the network parameters get tuned for all kinds of data and their variants and is expected to work on unseen or test data.

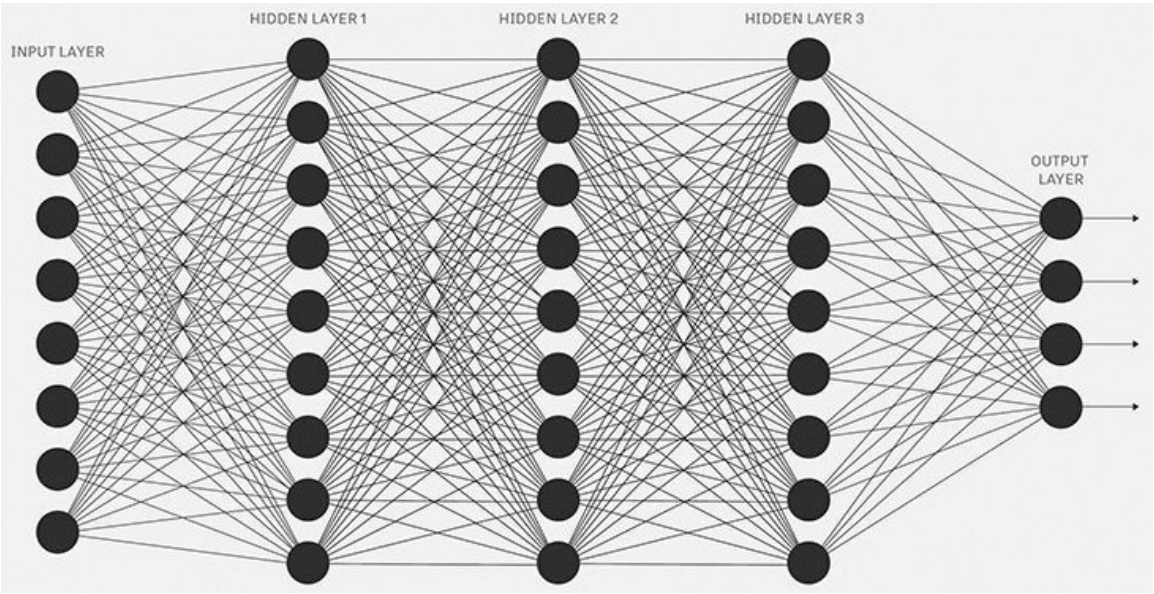


Figure 1.5: Typical architecture of an Artificial Neural Network [6].

In the past decade, this approach has been exceptionally successful for pretty much all kinds of problems. Furthermore, the simplicity of development made it the most popular machine learning choice with the research community. To clarify, all the stages such as preprocessing, feature extraction or classification are still happening inside the network, but are being crafted from the system itself

rather than by the developer's clever guidance. In the last few years, it has been almost decided that this technique works better than conventional machine learning for most possible problems in document analysis, including offline handwriting recognition.

The cost of this tremendous performance is paid in terms of the training time and a need for a massive quantity of data to be used. Deep learning in general is significantly slower than traditional machine learning when a model is being trained. This is usually compensated for by a technique called transfer learning. Rather than training an array of neural parameters initialized randomly, parameters from an already developed network are used as initial weights and the network is then trained for the current purpose. Networks such as AlexNet, VGG-16, VGG-32 and RESNET are very popular choices for transfer learning. Also, the data requirements (in both quantity and quality) are notably higher for Deep Learning. This can be compensated for by a method called data augmentation. Data augmentation is a technique which takes labelled data, modifies it (by stretching, skewing, rotating, adding distortions, etc.) to form a similar but different image of the same object and thereby generates a number of synthetic training data with known class labels which help increase the data volume required for Deep Learning. Even with these techniques, the overall training and data management process is distinguishably troublesome in this approach. Still it is widely used because of the simplicity and better end results. Furthermore, as the processors get faster with specialized GPUs and innovative tools appearing frequently, it is expected to continue being the most widely accepted toolbox in the future.

1.4 Introduction to the Bangla Writing System

Although the proposed methodology of offline recognition is applicable to almost any script, the fundamental technique and tools are developed to work with the Bangla script. Bangla, also called Bengali, is one of the most used languages in the world. With over 205 million people, it is the 7th most spoken native language. The Bangla script, used also for the Assamese language, is the fifth most widely used writing system in the world. It is the national and official language of the People's Republic of Bangladesh, and official language of several states in India such as West Bengal, Tripura, Assam and Andaman.

Bangla belongs to the Abugida class of writing systems. It is written from left to right. The script consists of 11 vowels, 10 vowel diacritics, 39 consonants, several hundred consonant conjuncts, more than 10 consonant diacritics, 10 numeric digits and several punctuation marks. There is no upper or lower case distinction of characters in the Bangla script.

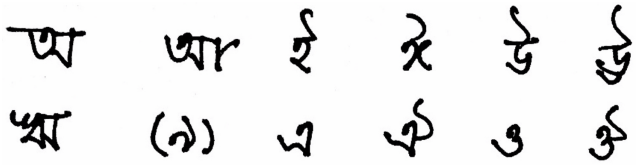


Figure 1.6: The Vowels in the Bangla Script.

Fig 1.6 shows the 11 vowels of the Bangla alphabet. The one in the bracket (2nd row, 2nd element) is not used anymore, but appears in historic documents. Fig 1.7 shows the 39 consonants of Bangla. The one in the bracket has double entries in the

alphabet by convention. The vowel graphemes with a consonant are used not as independent letters as shown in Fig 1.6, rather as diacritics attached to consonant letters. Fig 1.8 (a) shows an example of how this compares with the Latin script. Fig 1.8 (b) shows how the consonant 'Ma' appears with all possible vowel diacritics.

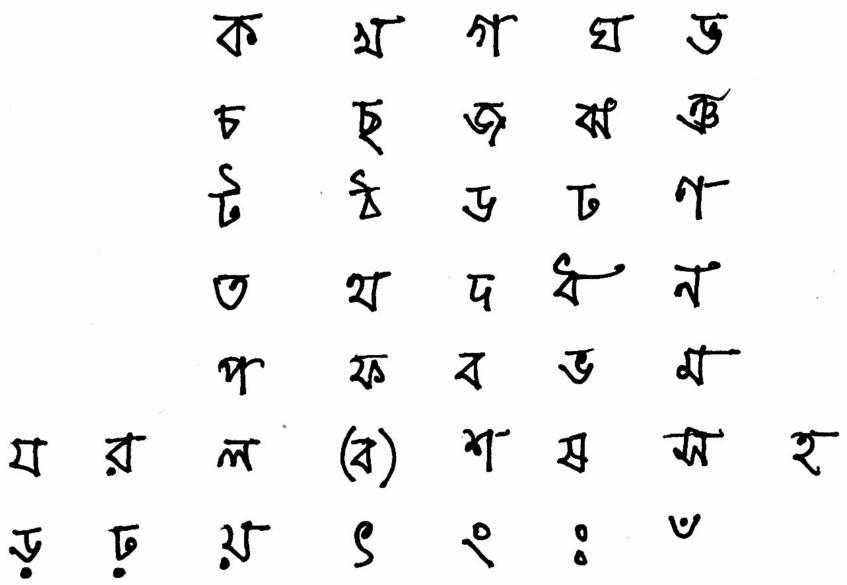
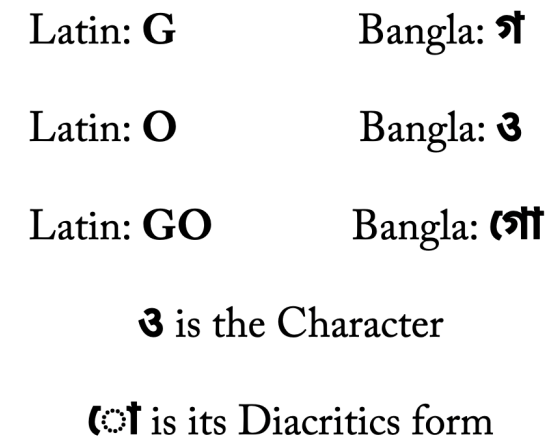
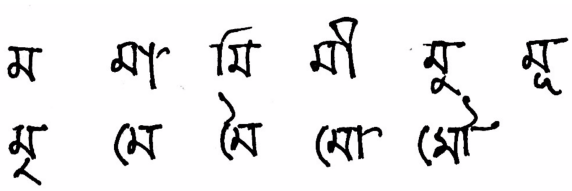


Figure 1.7: The Consonants in the Bangla Script.



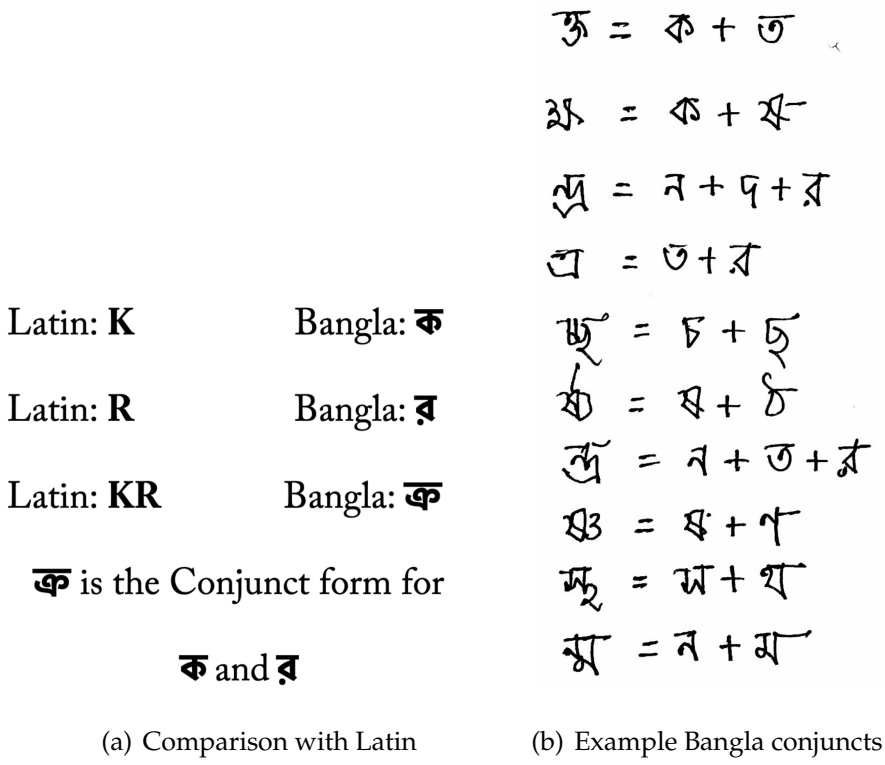
(a) Use of diacritic compared with Latin



(b) Vowel diacritics in Bangla

Figure 1.8: (a) Bangla diacritic use compared with the Latin script. (b) Diacritical forms of vowels with the consonant 'Ma'. The 'ô' sound is inherent to the solo consonant in the top left. Other vowel sounds are formed with the other diacritics attached to it.

When two or more consonants are adjacent without any vowel between them, they form a compound consonant or consonant conjunct and usually the form of the character is modified. An example comparing Bangla conjuncts with Latin is shown in Fig 1.9 (a). There are a few consonants which have their own diacritics while forming a conjunct like the vowels. Some of them have even more than one form of ligatures.



(a) Comparison with Latin (b) Example Bangla conjuncts

Figure 1.9: (a) Bangla consonant conjuncts compared with the Latin script. (b) A few consonant conjuncts in the Bangla script along with their component solo constants.

Fig 1.10 shows the Bangla numerals. The punctuation and other symbols are mostly similar to those used in the Latin script. One big and important difference is with the 'Period' symbol. It looks like '।', which is a little more distinct and makes the context based line identification process easier.

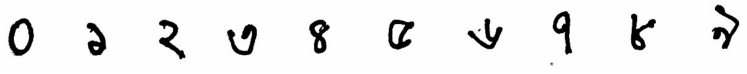
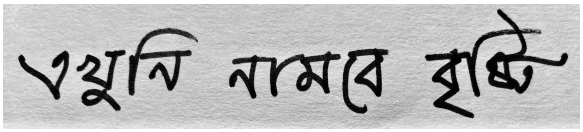


Figure 1.10: Bangla numerals, 0 to 9 from left to right.

One important attribute in Bangla and several other Indic scripts is the words are usually connected by a distinctive horizontal line running along the tops of the letters which can be seen in Fig 1.11 (a) and is highlighted in Fig 1.11 (b). This is known as a 'Matra', and this is very useful for character segmentation as will be discussed later. Also, the writing can be divided into three distinct zones which makes the recognition process systematic. The portion of the grapheme that appears above the "Matra" line is called the upper zone which can be due to the character symbol or vowel or consonant diacritics. From the Matra to the bottom-line (where all the characters end) is called the middle zone. Below the bottom-line there can also be vowel/consonant diacritics or some dots for distinctive consonants, this zone is called the lower zone. Fig 1.11 (b) demonstrates an example of how these zones are usually divided. All of these independent vowels and consonants from Fig 1.6 and Fig 1.7 are written from the same baseline, except for the last consonant in the alphabet, which is written on top of other characters when used. Many Bangla text recognition methods rely on locating these zones and identifying the components in each.



(a) Handwritten Bangla sentence

(b) Handwritten Bangla word

Figure 1.11: (a) A handwritten Bangla sentence, and (b) a word showing the matra, the base and the three zones.

1.5 Difficulties with Bangla Script for Offline Recognition

Bangla is a script with a large number of unique symbols compared to, for example, the Latin script. Furthermore, many characters and symbols look very similar to each other. Most complexities arise for this script because of the diacritics and conjuncts discussed in Section 1.4. Fig 1.8 (b) and Fig 1.9 (b) illustrates sample cases of Bangla diacritics and conjuncts and how they compare with the Latin script. As can be seen, both the diacritics and conjuncts increase the number of unique symbols while not increasing the alphabet. Diacritics can appear anywhere relative to the consonant as shown in Fig 1.8 and there are no rules for predicting the consonant conjuncts' physical appearances (shape or location) relative to the original letters. A conjunct can also have its own vowel diacritic. There can be up-to 5 different components inside such a structure. It is almost impossible to

separate these building block components from such a complicated structure and therefore these are better treated as unique symbols for machine learning. Because of these, the number of uniquely shaped glyphs (from a practical machine learning perspective) is more than 2000. For techniques like Deep Learning, there have to be hundreds (if not thousands) of labeled data samples for each class for training. Such a massive dataset is impractical to form just for one single script. Also, a network trained to recognize this number of classes cannot be expected to be fast or reliable. This is the fundamental reason why we haven't seen any noticeable progress in offline recognition of scripts like Bangla.

Furthermore, Bangla is a connected script by nature as can be seen from Fig 1.11 (a). Unlike Latin (or other scripts), this cannot be written in a way where characters in a word do not touch each other. This makes any kind of segmentation-based approach almost impossible to design. A few works that propose character segmentation techniques for Bangla only talk about segmenting a very small portion of the script, and also do not work very well. Also, while people write in vividly different manners, in Bangla there are several styles of writing which make many of the characters and conjuncts appear completely different. Offline handwriting recognition is itself a very complicated task, but with scripts like Bangla it is much more complicated. The only transcription works for this script reported so far recognize whole words out of a small vocabulary which can be useful for forms or fields where the expected number of input classes are restricted. Recognizing the whole script independent of vocabulary from handwritten images is an unsolved and almost untouched problem so far.

1.6 Scripts with Similar Properties

Bangla is an eastern Indo-Aryan language spoken in many parts of the Indian subcontinent. Lots of other scripts in this region share similar attributes with the Bangla writing system. The Assamese script is almost the same as Bangla except for some minor differences in a few letters. Some widely used scripts like Devanagari (for the Hindi language) and Gurmukhi (for the Punjabi language) also share close resemblance with Bangla. Hindi and Punjabi are the fourth and tenth most spoken first languages in the world. Many other Indo-Aryan scripts under the Abugida writing system also have close resemblance with each other. All of these scripts have a common root of the Brahmi script (which is no longer used). A list of such similar scripts, along with their orthographic properties is presented in Table 1.1 in alphabetic order. Fig 1.13 shows the map of Asia showing where the scripts from Table 1.1 are used. Color codes are used to indicate the amount of similarity from a holistic approach. A few handwriting samples of such similar scripts are shown in Fig 1.12. The attributes these other scripts share with Bangla are often so prominent that it is highly likely any recognition system developed for one of them should have a strong influence on, if not be exactly applicable, to the others.

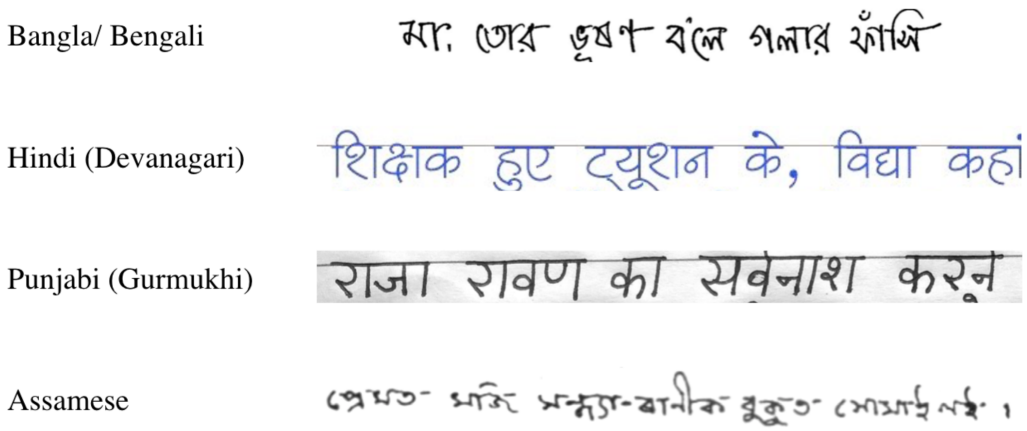


Figure 1.12: Handwriting samples of Bangla, Devanagari, Gurmukhi and Assamese scripts.

Table 1.1: Scripts with close resemblance to the Bangla writing system

No.	Name	Region	Language	Attributes	Other Notes
1	Assamese	Assam, Parts of Arunachal Pradesh and other northeast Indian states	Assamese/ Asamiya, Nagamese, Nefamese	• Written from left to right • Conjunct consonants • Diacritics for vowels • Horizontal line at top links the letters • Vowels appear as independent letters at the beginning of a syllable	• Originated from Kamarupi Prakrit • Alphabets are similar to Bangla except for one consonant
2	Bangla/ Bengali	Bangladesh, West Bengal, Assam, Tripura,	Bengali, Meithei, Bishnupriya Manipuri, Kokborok	• Written from left to right • Conjunct consonants • Diacritics for vowels • Horizontal line at top links the letters Diacritics for vowels • Vowels appear as independent letters at the beginning of a syllable	• Evolved from the Kamarupi script • Structural formation is less blocky and more of a sinuous shape - which is true for all the other scripts listed here

3	Devanagari	India and Nepal	Hindi, Marathi, Nepali, Maithili, Bhojpuri, Dogri, Rajasthani, Chhattisgarhi, Santali, Newar, Kashmiri, Konkani, Sindhi, Bodo, Awadhi, Magahi, Haryanvi, Bhili, Mundari, Sanskrit, Pali	• Written from left to right • Conjunct consonants • Horizontal line at top links the letters Diacritics for vowels • Vowels appear as independent letters at the beginning of a syllable	• Has evolved from ancient Brahmi script • 4th most used writing system in the world • Over 120 languages use this script
4	Gurmukhi	Punjab region (India, Pakistan)	Punjabi language, Sant Bhasha, Sindhi language	• Written from left to right • Conjunct consonants • Horizontal line at top links the letters Diacritics for vowels • Vowels appear as independent letters at the beginning of a syllable	• The script is very similar to other Indic scripts, except for angles and structural emphasis. • In the top 15 writing systems in the world
5	Gujarati	Punjab region (India, Pakistan)	Punjabi language, Sant Bhasha, Sindhi language	• Written from left to right • Conjunct consonants • Horizontal line at top links the letters Diacritics for vowels • Vowels appear as independent letters at the beginning of a syllable	• The script is very similar to other Indic scripts, except for angles and structural emphasis. • In the top 15 writing systems in the world

6	Gujarati	Gujarat, Rajasthan, Maharashtra, Madhya Pradesh, Karnataka, Pakistan, Eastern Iranian Plateau	Gujarati, Sanskrit, Kutchi, Avestan, Bhili, Dungra, Bhil, Gamit, Chowdhary, Kukna, Rajput Garasia, Varli, Vasavi	• Written from left to right • Conjunct consonants • Horizontal line at top links the letters Diacritics for vowels • Vowels appear as independent letters at the beginning of a syllable	• Used to write the Gujarati and Kutchi languages • Does not use Matra.
7	Khmer	Cambodia and Thailand	Khmer	• Written from left to right • Diacritics for vowels • Consonant cluster, a variant of conjunct consonant - where the 2nd consonant is written under the main consonant in reduced form	• Words are not separated by spaces • There are some independent vowel signs, which appear in relatively few words • Not all consonants can be at the syllable final position
8	Khudabadi	Sindhis in India	Sindhi language	• Written from left to right • Diacritics for vowels • Matras are used to change the inherent vowel. • Vowels that appear at the beginning of a word are written as independent letters. • Special Conjunct symbols are used	• Also known as Vaniki, Hatvaniki and Hatkai script.

9	Modi	Maharashtra, India	Marathi (primary), Konkani, Gujarati, Hindi, Kannada, Telugu, Tamil, Urdu and Sanskrit	• Written from left to right • Diacritics for vowels • Head-strokes appear before the letters • Consonant conjuncts	• Derives from Nagari family of scripts • Cursive feature makes it very useful for shorthand
10	Odia/Oriya	Odisha, India	Odia, Sanskrit	• Written from left to right • Diacritics for vowels • Vowels that appear at the beginning of a word are written as independent letters. • Conjunct letters	• Developed from the Kalinga Alphabet • Noticeable similarities with the Thai alphabet
11	Ranjana/ Lanydza	Nepal, Buddhist monasteries in India, China, Mongolia, and Japan	Newar, Sanskrit (in Tibet), Tibetan	• Normally written from left to right • Diacritics for vowels	• Known as Nepali calligraphic script • In Kutakshar form, written from top to bottom
12	Sarada/ Śāradā/ Sharada	Kashmir	Sanskrit, Kashmiri	• Written from left to right • Diacritics for vowels	• Origin of the Gurmukhi script
13	Syloti Nagari	Sylhet, Mymensingh, Netrokona, Kishoreganj of Bangladesh and in India's Assam	Sylheti language, Bangla language	• Written from left to right • Horizontal line at top links the letters Diacritics for vowels • Vowels appear as independent letters at the beginning of a syllable	• Alphabets are a subset of Bangla • Doesn't have any ligatures

14	Tagbanwa	Philippines	Languages of Palawan	• Usually written from left to right in horizontal lines. • Diacritics and independent symbols for vowels	• Syllables ending in a consonant are written without the final consonant • Traditionally written on bamboo in vertical columns from bottom to top and left to right
15	Tibetan	Tibet, Bhutan, India, Nepal	Tibetan, Dzongkha, Ladakhi, Sikkimese, Balti, Tamang, Sherpa, Yolmo, Tshangla, Gurung	• Written from left to right • Diacritics for vowels • Consonant clusters	• The printed form of the alphabet is called the Uchen script • The handwritten cursive form used in everyday writing is called the Umê script. • Spaces not used to divide words
16	Tirhuta/ Mithilakshar	Mithilia region of Bihar, India and the eastern Terai region of Nepal.	Maithili, Sanskrit	• Written from left to right • Conjunct letters • Diacritics for vowels	• Considered a sister system of the Bangla writing system • Most of the letters and numerals are identical

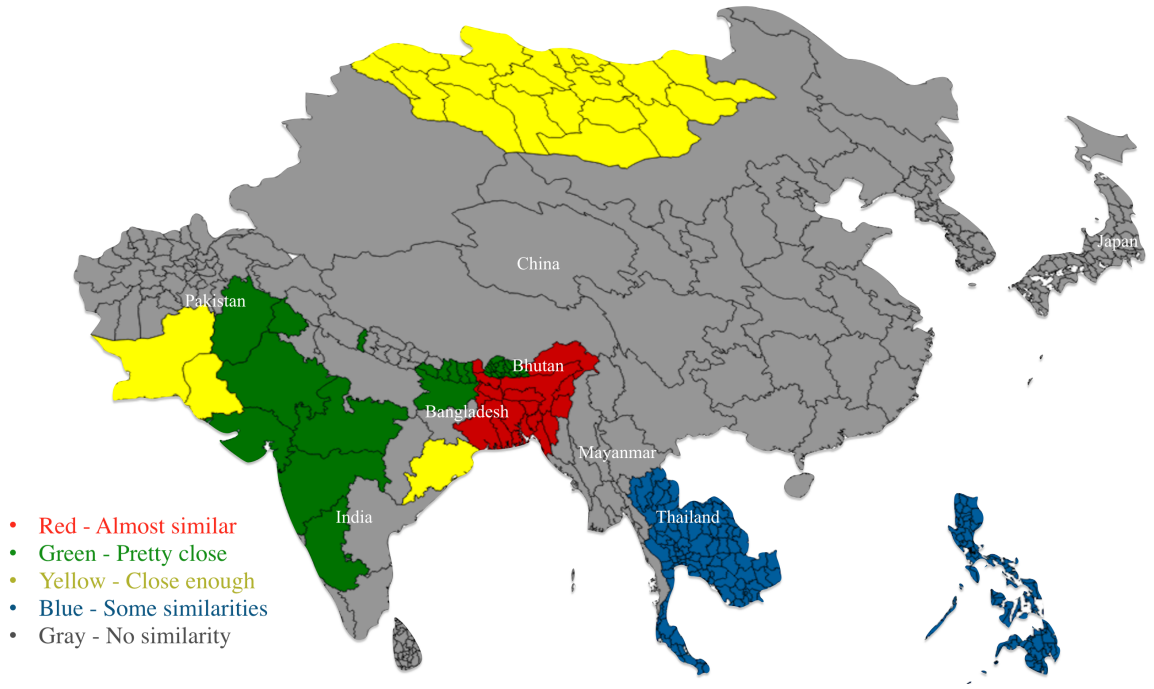


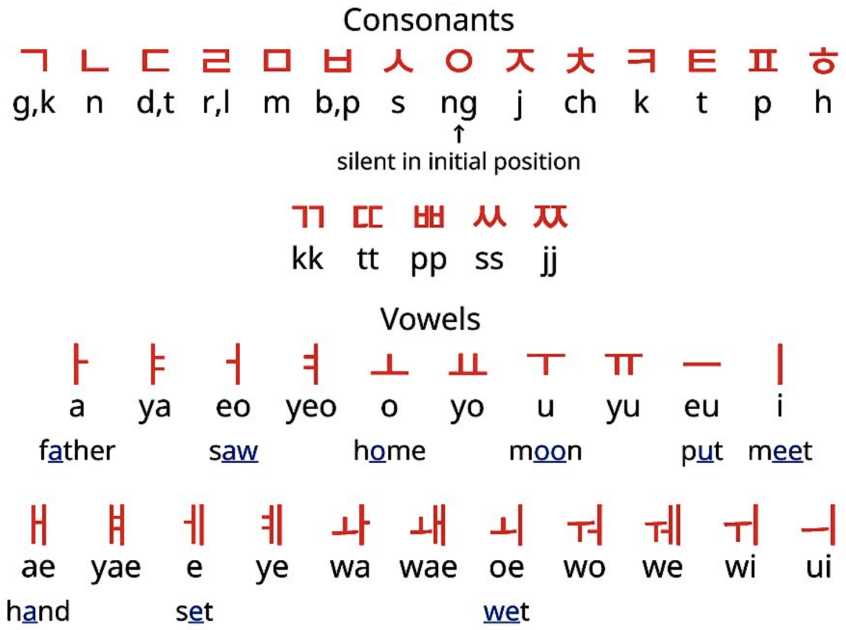
Figure 1.13: Map of Asia, with colors representing the similarity of the scripts to Bangla.

1.7 Introduction to the Korean Script

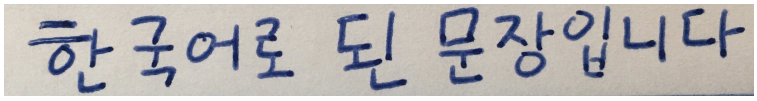
The Korean script is known as Hangul/Hangeul in South Korea and Chosŏn'gŭl in North Korea. This is the official script of Korea and a co-official writing system in the Yanbian Korean Autonomous Prefecture and Changbai Korean Autonomous County in Jilin Province, China. Some parts of Indonesia and Taiwan also use this script. There are 28 basic characters or Jamos in the Hangul alphabet. These are arranged into syllables by combining up-to 6 characters. This syllabic arrangement happens in two dimensions, i.e. they can combine both horizontally and vertically which is one of the most distinguishing features of Hangul. However, unlike

Bangla, the shapes of the original characters don't change based on the character sequencing process. This particular property makes Korean an easier script to process than Bangla from a machine learning perspective. Furthermore, Hangul characters are written separately and are not inherently connected like Bangla or cursive Latin, although they do often touch when handwritten.

The Korean alphabet and a handwriting sample is shown in Fig 1.14. The Hangul syllable is composed of three constituent parts: lead consonant, vowel, post-consonant. The top left is the first consonant. If the vowel is a vertical symbol that would appear on the upper right of the syllable, a horizontal symbol would appear below the lead consonant, or a compound vowel would appear in both geometric places. The post-consonant (if the syllable includes one) follows at the bottom.



(a) Korean alphabet [7]



(b) Korean handwriting sample

Figure 1.14: The alphabet (top) and handwriting sample (bottom) of the Korean/Hangul script.

1.8 Outline of this Dissertation

Offline handwriting recognition is considered to be an unsolved problem. The traditional techniques and notable achievements for different scripts are discussed in Chapter 2. Chapter 3 introduces the presented offline recognition design along with the underlying tools and technologies which make it reliable and flexible. This framework is thoroughly tested with the Bangla script and implemented for

the Korean script to demonstrate its adaptability. Chapter 3 also describes many other supporting experiments for this framework and tools developed for this research. Chapter 4 presents the results along with comparison with other related works (when present) for all the experiments described in Chapter 3. Finally, Chapter 5 concludes by summarizing the contributions of this research as well as discussing the potential future direction.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview: It's an Unsolved Problem

Over the years, there has been a lot of research and development in offline recognition, but still today it is considered to be an unsolved problem. Different scripts have seen different amounts of progress. For example, popular scripts like Latin have seen a lot more attention in the research community than most Abugida scripts. Many approaches in practice today are script dependent and there is no unified framework that works for all. For Bangla, even though it is one of the most widely used scripts in the world, the offline recognition progress is close to none. There are a lot of scattered works and solutions have been presented for some specific problems under certain conditions, but the broader aspect of transcribing from unconstrained Bangla handwriting is yet to be reported. In other words, there are no works published which can dependably recognize Bangla handwriting without vocabulary restriction. Most of the works are based on isolated character/number recognition, character segmentation algorithms, whole word recognition from a limited sized word list and word spotting algorithms. Since many Abugida scripts listed in Table 1.1 are very similar to Bangla, the following subsections discuss the

recent and notable algorithms and achievements for Bangla as well as these other scripts. Also, the availability and conditions of Bangla datasets which are publicly accessible for researchers are included in this discussion.

2.2 Segmentation-Based Approaches

As discussed in Section 1.3.1, the process of character segmentation can be extremely difficult to achieve cleanly. This is true for almost every script and Bangla is by no means an exception. Handwriting recognition is most challenging when the characters are connected like Bangla, cursive Latin, Arabic etc. While segmentation-free methods can be very scalable, flexible and script independent, in contrast, the process of segmentation is usually different for different writing systems. Most of the time, this depends on the core attributes and minute details of the concerned script. Bangla (and similar scripts) have a relatively large number of graphemes when the diacritics and conjuncts are considered. One of the key traits that often has been exploited for character segmentation is the presence of the “Matra” line in Bangla words. The "Matra" is the horizontal line shown in Fig 1.11, on top of the characters to connect them in a word. Although not all letters in a word have this, in most cases it is present. The trail of this line is easy to track even in quick or bad handwriting. Usually, this is the line that connects adjacent characters (or diacritics) together and if this line were removed, the words appear to be a combination of isolated characters, closely but separately compiled together.

Pal and Dutta used this property with a concept based on the water reservoir

principle to segment unconstrained Bangla handwritten text [8]. Here, virtual reservoirs are formed by pouring water from the bottom of the words, and due to that horizontal line "Matra", a large accumulation is found in the middle zone between the characters as shown in Fig 2.1. Furthermore, their observation showed connected characters have a larger number of reservoirs than isolated characters, and vertical overlapping of a reservoir with a loop or another reservoir doesn't happen in isolated characters, whereas it does happen frequently with connected characters. By exploiting these detailed observations they succeeded in obtaining a result with very high accuracy. Most of the error in their work came from the connected characters identified as isolated. By further considering the script specifics, even more improved results might be achievable in this process. Later they used this approach of segmentation to recognize city names (84 classes) using Dynamic Programming, MQDF and directional features achieving recognition accuracy between 87% and 94% for different arrangements [9, 10]. This work was primarily targeted at Indian postal automation and works on a heavily restricted vocabulary.

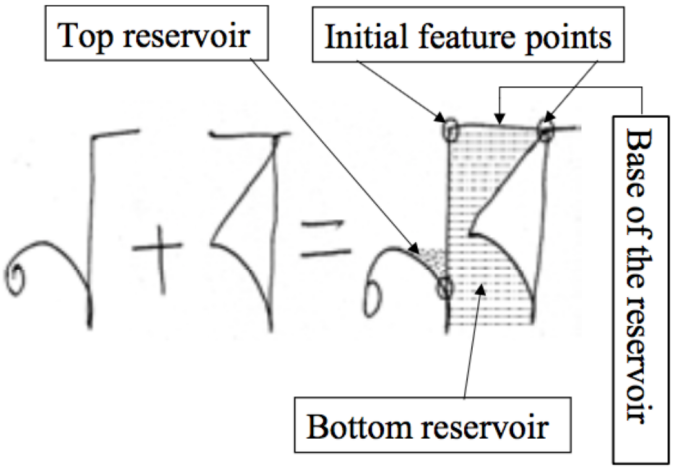


Figure 2.1: Demonstration of large bottom reservoir usually found in the joining section of two characters in a Bangla handwritten word [8]

.

The use of the "Matra", as found in many Indo-Aryan scripts, often provides a simple answer to Sayre's paradox [5], i.e. segmentation without prior knowledge about the characters. Numerous works have reflected this idea recently. Basu et al. presented a fuzzy technique to identify the black pixels in the Matra and then marked points on that line as segmentation points [11]. Identifying the Matra line was done by a simple sum of the row method to find the longest run of dark pixels. Afterwards, each dark pixel was associated with a region of the Matra based on its proximity using fuzzy logic. The segment points on the Matra were computed on the basis of a fuzzy membership function. This approach was further improved [12, 13] by detecting the zones in a word (as shown in Fig 1.11) to locate the Matra efficiently and using Multilayer Perceptron (MLP) classifiers to identify whether the segmented image does or does not need further segmentation. Segmenting in this way, where the goal is to ensure at least one character stays inside the box, is

known as under-segmentation. The counter way is known as over-segmentation, which attempts to fit no more than one character in each segment and allow characters to be split into multiple partitions.

In handwriting, the Matra might not always be a straight and/or horizontal line which can cause issues for the techniques developed based on this. Roy et al. [14] addressed this problem and proposed an intuitive skew detection and correction routine to use before segmenting characters using zone oriented connected components. Bhowmik et al. presented an area-based algorithm based on the Hough Transformation for skew adjustment and afterwards used an MLP to classify the contour points as to whether or not they are a break point between characters [15]. An interesting way of dealing with skew issues was proposed by Bag et al. [16]. They used vertex characterization of the outer isothetic polygonal cover (for a word, or part of a word) to segment characters without the necessity of de-skewing. Another frequent issue that troubles the segmentation process is the overlap between rectangular hulls of successive characters as addressed by Bishnu and Chaudhuri [17]. They proposed a recursive contour following one of the zones along the height of the word to locate the extent where the major portion of the character gets covered. In a more recent work, Bhattacharyya and Sarma [18] used an Artificial Neural Network (ANN) combined with isolated character recognition to develop a character segmentation method for the Assamese script, which is mostly the same as Bangla. They first used a horizontal projection histogram to perform primary segmentation. Afterwards, the segmented images are matched with machine printed characters using a MultiLayer Perceptron (MLP) network. If a match is found, then the segmentation is considered to be correct, if

not then the segmentation boundary is moved until it finds a match. Although this work doesn't properly address the conjuncts or vowel combinations, the method obtained over 95% accuracy for a limited character class.

Numerous similar attempts were made for other Abugida scripts. For example, Tripathy and Pal used a scheme based on the water reservoir approach [19] on unconstrained Oriya script. Though Oriya shows a lot of similarities with Bangla in many aspects, it doesn't feature a Matra. The connection point of two Oriya characters were estimated using some frequent scenarios - (i) consecutive characters create a large bottom and a small top reservoir (near the mean line), (ii) the number of reservoirs and loops are greater in connected characters and (iii) the morphology of touching characters are more complex than isolated ones. In contrast, the Devanagari script features a Matra the same way as in Bangla, and once again this appears very useful in character segmentation. Garg et al. [20] presented a method for segmenting Hindi text by first identifying the three zones (equivalent to Bangla as in Fig 1.11), then locating the Matra (the line with the highest number of dark pixels in the upper zone) followed by looking for zero or a minimum number of dark pixels in a vertical projection from the Matra to the middle zone. In another attempt, Hanmandlu et al. [21] worked on removing the Matra from a word for Hindi text and thus all the characters get separated on their own in the middle zone. This is equally true for the Bangla writing system; i.e without the "Matra", individual characters become isolated, becoming disconnected from their neighbors.

Expanding the framework of segmentation for consonant conjuncts is possible, but is often avoided due to its enormous level of complexity, rather it is considered

to be convenient to treat these compounds as unique objects. There are a few works reported for the Devanagari script [20, 22] which attempt to segment fused consonants. These algorithms should be transferable to Bangla with little to no modification due to the characteristic similarity of these scripts.

A clever way of tackling OHR is through N-gram modeling, which requires segmentation not into individual characters, rather into N adjacent characters. N is usually 2 (bigram) or 3 (trigram). The biggest OHR issue comes from the unprecedented variability for depicting characters, even with the same person from the same page; and statistically speaking, there is more similarity in how people write a set of characters than how they write individual characters, since the neighboring characters influence the writing process. The N-gram modeling takes advantage of this characteristic of handwriting and is often used to improve the recognition result as a second pass iteration. This approach is very useful for the scripts with fewer characters. For example, in the Latin script with 52 character symbols, there can be 52×52 bigram patterns, of which many are seldom or never used in common text. Things are much more complicated for scripts like Bangla where there are more than a thousand different symbols (due to the consonant conjuncts and diacritics, where the conjuncts are already bigrams, trigrams or quadrigrams) [23]. This makes the model synthesizing process extremely complicated and impractical.

In theory, a character segmentation and an isolated character recognition process can be merged into a full transcription unit, although no one has reported such a complete system. There are many works in isolated Bangla character recognition that yield great success. As discussed before, these usually go through some form

of feature extraction and then a choice of classifier. The feature extractor attempts to obtain useful structural features of the glyph, such as height, aspect ratio, geometry of strokes, edge points, lines and loops, connection between shapes, and to describe them in compressed forms like, histograms, color maps, vectors etc. This makes the classification and training process simple, coherent and computationally efficient. Choosing the right set of distinct and independent features is very important for classifier performance, which makes this process sensitive to script specifics. There are two core approaches for feature extraction - statistical and structural. Statistical approaches usually require more data and computational resources than structural approaches, but provide better immunity to noise and distortions [24]. The structural approaches are very intuitive to how the human mind works, and offer better tolerance to the variation of natural handwriting [25], but preparing such an architecture with a confined rule base is often very challenging and problem dependent. There are also hybrid approaches, where both statistical and structural processes are adopted for the same problem. Usually, only a few relevant and distinctive features from all the extracted ones are presented to the classifier - a process known as feature selection. From these, the classifier creates an abstract class/model of the symbols which is used in training, and ultimately decision making. Lots of classifiers have proved useful in handwriting recognition, such as Support Vector Machines (SVM), Hidden Markov Model (HMM), K Nearest Neighbor (KNN), Bayes classifier etc. Some are particularly tailored to NN models, such as Hopfield Network, MultiLayer Perceptron (MLP), Back Propagation (BP), Deep Learning (DL) based classifiers etc.

Bhowmik et al. presented a Hierarchical Classification Architecture with SVM

for Bangla character recognition [26], and compared the results with a MultiLayer Perceptron (MLP) and a Radial Basis Function (RBF) network. The Hierarchical Learning Architectures (HLA) based SVM outperforms the others, although a fusion scheme of all these three classifiers was also proposed which turned out to be marginally better than the SVM alone. Features were extracted using the Daubechies wavelet transformation. For an image the wavelet transformation generates one low frequency sub-band approximation and three high frequency components for the details of the original image. The feature vector is prepared from the approximation component for different resolutions.

The other two classifiers used, MLP and RBF, belong to the class of Neural Networks. MLP has been a very popular choice for handwriting recognition. For the task in [26], a modified Back Propagation (BP) model with self adaptive learning rate values was used for the training. RBF is also widely used in many pattern recognition problems, if not specifically for handwriting recognition. In the network architecture, a Gaussian activation function was used as the basis function and outputs are augmented by a sigmoid function. The gradient decent learning method was applied to tune the network weights and basis function parameters during supervised learning. When several classifiers are used for a single problem, there are usually several approaches that can be implemented. In a basic single stage classification scheme, each of the classifiers is implemented independently and separately. In a single stage fusion scheme, the decision is taken upon majority votes from the results. In a Hierarchical Learning Architecture (HLA), initially the samples are grouped based on their basic features, and then each group is processed by a particular classifier which had the best success for that group during

training.

In terms of multilayer hierarchical classification, Reza and Khan proposed a method for grouping similarly shaped basic characters, numerals and vowel modifiers [27], and used the One Versus One (OVO) and One Versus All (OVA) strategies for multi-class SVM for better performance. They used a Zonal Directional Chain Code for feature extraction and compared the grouping results with morphologically thickened and thinned segmented text. This kind of grouping almost always benefits the classification process. As for the classifier, the SVM is one of the most popular choices in the field of OHR for its simplicity and efficacy. Majumdar and Chaudhuri used a OVA SVM with features extracted from the Curvelet Transform [28]. They worked with four versions of morphologically thickened and thinned samples and presented the comparison results. Das et al. used a MLP and SVM for classification of Bangla simple and compound characters [29]. They included 55 of the consonant conjuncts based on their high frequency of occurrence in common literature. A single hidden layer MLP was used with a quad tree-based shadow and longest run feature for group identification. Later on they used a Genetic Algorithm (GA) and SVM in a multistage approach for recognizing these compound characters [30] using both global and local feature extractors. Also, they proposed a two-pass approach along with an automated grouping of characters in coarse classification [31] for a signifiant jump in accuracy. They used a GA to select the local region optimally to extract features with potential finer classification, a Quad tree based Longest Run for features and an SVM as the pattern classifier. The same team also presented a benchmark image dataset for isolated Bangla compound characters [32] available publicly for use by other researchers.

A combination of MLP and SVM is also used by Mohiuddin et al. for online Bangla handwriting recognition [33]. Ahmed et al. used an SVM for Bangla handwritten numeral recognition (from online samples) combined with a Supervised Locally Linear Embedding (SLLE) algorithm [34]. Their approach presented a system which doesn't rely on massive training. Wen et al. proposed two approaches for handwritten Bangla numeral recognition [35] – one uses image reconstruction based on Principle Component Analysis (PCA) targeted for reducing the difference between the same symbols during preprocessing, and the other uses direction feature extraction from a Kirsch edge detector based on a combination of PCA and SVM. One more interesting and relevant application of a SVM can be found from work by Pal et al., where they used Gaussian Grid feature extraction for offline Bangla signature verification [36].

The Hidden Markov Model (HMM) is a powerful tool in the field for handwriting recognition, not only for holistic approaches as discussed earlier. Roy et al. proposed a zone segmentation based OHR for Bangla [37] using a HMM in the middle zone for character recognition, and a SVM in the upper and lower zone for modifier identification. They used a Local Gradient Histogram (LGH) as the feature extractor and combined the results from the different zones forming the word as a whole. This produced results with much greater accuracy than an HMM alone for a sole segmentation-free word recognition. Another approach from the work of Bhowmik et al. [38] uses a MLP based on stroke features. They extracted vertical and horizontal strokes and saved them as one-pixel thick digital curves. Afterwards, they extracted features such as shape and size from these skeleton strokes and fed them into a MLP network. This is a kind of structural approach that

can be useful for other scripts too. A relatively old work by Dutta and Chaudhury also demonstrates the power of stroke features, such as curvature, where junctions can be exploited for both machine printed and handwritten characters [39]. Bag et al. presented a list of structural approaches used for Bangla offline handwritten character recognition, along with proposing a method which is invariant to the angle of observation [40]. They used concavity/convexity of character strokes realized in a skeleton format as features for classification.

Much research work has been reported for the scripts listed in Table 1.1, especially for the major ones. Arora et al. presented a combination of four feature extraction techniques [41] - intersection, shadow, chain code histogram and straight line fitting with a simple feed forward MultiLayer Perceptron classifier for handwritten Devanagari character recognition. Shadow features were computed globally for the character, where the other three were applied on different small segments. This approach can also be applied to recognition of other scripts, such as Bangla as anticipated by the authors. Sharma et al. proposed chain code based features combined with a Modified Quadratic Discriminant function (MQDF) classifier for Devanagari simple characters and numerals [42] - which also offers the potential for other similar scripts. Sharma and Jhajj developed a zoning method for feature extraction and compared the results with KNN and SVM classifiers for handwritten Gurmukhi script [43]. From their work, a SVM with a polynomial kernel gives the best results. They also described potential reasons for failure, which can be useful for further research in this domain. Kumar et al. used a KNN classifier for Gurmukhi, where features were extracted with diagonal and transition features [44]. Hassan et al. presented a Multiple Kernel Learning (MKL)

based classifier embedded in a Decision DAG framework for Gujarati character recognition [45]. They compared their results for three different feature extractors – fringe feature maps, shape descriptors and Histogram of Oriented Gradients (HOG) – with MKL 1-vs-1 and KNN classifiers.

As mentioned in Section 1.3.4, Deep Learning is the modern and most successful approach for almost any problems related to machine learning, and this statement is applicable to isolated character recognition as well. Although, some people tend to avoid Deep Learning since training time and the required quantity of data are much greater than traditional machine learning. Still, because of the simplicity in the architecture with no pre-processing, feature extraction or classifier design, it is one of the most popular choices among researchers. One notable work is done by Alif et al. [46]. They used a modified Residual Network (ResNet-18) architecture to classify isolated characters with 95.10% recognition accuracy. This is the highest reported accuracy for a work for Bangla which addresses a few high frequency conjuncts. In fact, using Deep Learning isolated character recognition for any script is now considered to be a solved problem if enough data is available. This is because of the "near-human performance" obtained with the MNIST dataset (handwritten latin digits) using neural architectures. Several researchers achieved over 99.97% [47], [48], [49] with this dataset in the last decade.

Devanagari is a script which has close resemblance to Bangla in many aspects. Dutta et al. presented a work with this script [50]with a CNN-RNN hybrid architecture and lexicon-based decoding on the IIIT-HW-Dev dataset. Most other works with Indic or similar scripts are restricted to either printed or online documents for word-level recognition. Some scripts like Korean are inherently isolated and

the syllables do not touch each other (except accidentally) while written. Offline recognition of such scripts is usually achieved by a feature extractor and a classifier, or only a classifier if using a Deep Learning architecture. Park et al. presented an implementation using fifteen character normalization, five feature extraction and four classification methods and evaluated their performance on two public handwritten Hangul datasets, SERI and PE92 [51], and obtained 93.71% and 85.99% syllable recognition accuracy respectively [52]. Kim and Xie used Deep Learning based modeling on the SERI95a and the PE92 datasets and achieved 95.96% and 92.92% recognition accuracy respectively, which is the highest reported for Korean [53]. Most other approaches reported for the Korean also follow either of these two structures for offline recognition. Technically these are segmentation-free frameworks as Korean script is already segmented by nature, but the approaches are better categorized as symbol recognition which matches with the segmentation-based methods (once segmentation is done) and therefore are presented in this section.

2.3 Segmentation-Free Approaches

Only a handful of works have been done using segmentation-free approaches for Bangla and similar scripts. Broadly, there are two categories of work in this approach so far – word spotting and limited vocabulary based word recognition. For segmentation-free word spotting, one of the most flexible ideas was proposed by Rothacker et al. [54]. They used a Bag-of-Features (BoF) representation powered with SIFT [55] descriptors to feed a Hidden Markov Model (HMM). SIFT is a powerful algorithm which extracts and describes local features from an image.

Many features can be generated for even a small object with high efficiency (close to real-time performance). This is invariant to image scale and rotation, and offers robustness to affine distortion, noise, illumination variance and skewing. All of these make SIFT very useful in object recognition including handwritten documents. In their work, the codebook was prepared using a Generalized Lloyd [56] clustering algorithm on a randomly sampled portion of all descriptors. In fact, as a follow-up to this process, Sebastian Sudholt et al. presented a pipeline of descriptor learning for word spotting [57]. Here the objective was to find descriptor pairs which are closely spaced and far apart in the descriptor space to reduce the errors originating from quantization. For the work in [54] the model estimation was done by a bounding box query around a word to get a Bag-of-Features (BoF) HMM encoded sequential visual appearance. After a patch-based query from the document collection (they used the George Washington dataset of cursive English/Latin document), the Viterbi algorithm was used to decode the model.

Bag-of-Features is a very effective approach to find distinct local histogram signatures of objects from a scene. This converts vector-represented patches to codewords, and prepares a codebook from those, like a dictionary. A codeword can be thought of as being representative of several similar patches. Combined with an HMM, one of the most powerful statistical tools in handwriting recognition (both online and offline), their work produced excellent results. Although one of the most incredible outcomes of this approach comes from its flexibility, as the work produced excellent results not only with the George Washington dataset, it also came out to be very successful with Arabic [58] as well the Bangla script [54] - without being specifically tailored for those. An overview of the Rothacker et al.

work with the Bangla printed text is shown in Fig 2.2. The flexibility of this method makes it very promising for handling different kinds of scripts. Specifically, it is very useful for Bangla and similar Indo-Aryan scripts from Table 1.1, because the scripts' inherent nature makes it difficult to dissect the characters from a word as discussed in Section 2.2. Word spotting algorithms find a word based on its shape. The textual content of that word has to be stored external to the query. Only whole words are usually queried, so this limits the method to a vocabulary, and the process has to be repeated for each possible word. These algorithms always need to have a prototype word in the document.

Figure 2.2: Word spotting overview of work by Rothacker et al. with Bangla printed text [54].

Another interesting method that does keyword spotting without segmentation from Bangla handwriting was proposed by Zhang et al. [59]. They pre-

sented several popular ways of detecting keypoints such as SIFT, LoG (Laplacian of Gaussian), the Harris algorithm, basic Morphological operations, etc., with a nice comparative illustration as shown in Fig 2.3. In their work, the SIFT detector and morphological operations were used for detecting keypoints. The patch size was dynamic, decided by the local entropy around the keypoint. The Heat Kernel Signature (HKS) was used as a fundamental feature descriptor and it was shown to be more variation tolerant than SIFT features. The computational efficiency was increased by narrowing the searching scope for keywords to only within the zones which initially have enough matching of the keypoints.

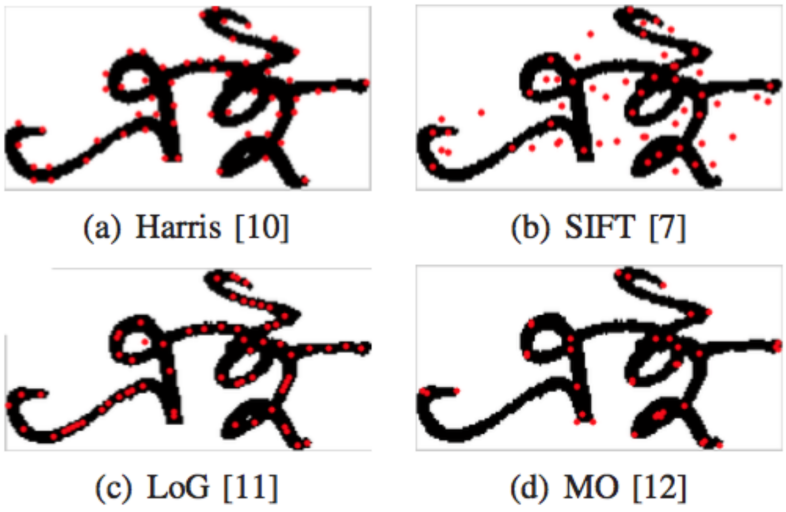


Figure 2.3: Detection of keypoints from a Bangla handwritten word using different algorithms [59].

Wshah et al. proposed a way of script independent word spotting from offline handwritten documents [60] using an HMM which produced great results with the English, Arabic and Devanagari scripts. This is a line-based approach backed by both character-based and lexicon-based background frameworks. A similar

system was proposed by Das et al. [31] for Indic scripts highlighted for Bangla and Devanagari, where Pyramid Histogram of Oriented Gradient (PHOG) was used as the word spotting framework combining foreground and background features. A comparative study [61] shows that PHOG is better suited than other features for Bangla Handwriting, especially on the middle zone. Shekhar and Jawahar worked on word image retrieval [62] using Bag of Visual Words (BoVW) for four important Indian scipts including Bangla. Here, the interest points were computed with the Harris Corner Detector and SIFT was used to describe the local information at those interest points. The retrieval was done from the histogram index structure in a ranked manner using a Lucene search engine. One of the major issues with BoVW is that it ignores the spatial relationships between visual words, which was addressed by repeatedly subdividing the image and getting local features in finer resolution with Spatial Pyramid Matching (SPM). In another work Vajda and Belaïd proposed a Non-Symmetric Half Plane Hidden Markov Model (NSHP-HMM) for segmentation-free handwriting word recognition [63] working for both Latin and Bangla. The technique applied here was to implant high level structural information by a weighting mechanism in the baseline NSHP-HMM.

There are a group of works which approach Bangla whole word recognition by segmentation-free approaches, but all of these are restricted to a limited vocabulary. For example S. Bhowmik et al. reported several different approaches in two different papers to recognize whole words. They used an SVM and MLP along with other histogram based features and operated on 18,000 words with 120 different classes and obtained 83.64% recognition accuracy [64]. In another work, they used a Neural Network with HOG features on 1020 words with 20 different

class and achieved 87.35% accuracy [65]. T. K. Bhowmik et al. used an HMM and Genetic Algorithm (GA) on shape based direction features [66]. They obtained a Word Recognition Accuracy (WRA) of 79.10% on a 35,700 word image dataset with 119 different classes.

As mentioned, nothing reliable has been reported in terms of vocabulary independent transcription for Bangla or any of the scripts from Table 1.1. In a vocabulary dependent framework, Adak et al. applied their approach on a relatively small but random Bangla word dataset [67] and obtained a WRA of 70.67%. This is the only attempt for unconstrained Bangla text recognition prior to my work. Adak et al. also worked on restricted vocabulary based recognition with several combinations of datasets and got a WRA of close to 90% in many cases.

The most common approaches of segmentation-free word recognition usually use either the Long Short-Term Memory (LSTM) or the HMM architecture. An LSTM is a form of a Recurrent Neural Network (RNN) which maintains a memory using self connecting loops to prevent a vanishing gradient from affecting distinct information. One big advantage from HMMs or other RNNs that makes LSTM suitable for these kind of problems is its relative insensitivity to gap length. In fact, some researchers think it may be possible to build a universal language independent OCR using such networks [68]. Most successful works using such architectures can be found for purely alphabetic scripts like Latin, Arabic, etc. For example, Bluche et al. used the ROVER (Recognition Output Voting Error Reduction) scheme to combine four models and reported results on the IAM (English) dataset [69]. Two of them are based on Bidirectional Long Short-Term Memory (BDLSTM) RNNs, and the other two are based on deep Multi-Layer Perceptrons

(MLPs). Menasri et al. proposed a system that uses seven recognizers based on three different technologies - grapheme based hybrid HMM, Gaussian Mixture HMM and Recurrent Neural Networks (RNN) [70]. They presented their results with the RIMES (French) dataset. Stahlberg and Vogel proposed a method that uses fully connected deep neural networks for optical modeling with features extracted from raw pixel gray-scale intensity values of foreground segments [71] and presented their results with the IFN/ENIT (Arabic) dataset.

LSTM networks were used for whole word [72], isolated component [73], machine printed [74] and online handwriting recognition [75], but never for offline Bangla transcription. A similar situation exists for other Indic scripts too, such as for the printed Devanagari script Karayil et al. used (LSTM) networks [76] and Sankaran and Jawahar used Bidirectional LSTM (BLSTM) [77] and obtained over 80% word recognition accuracy. None has reported LSTM or HMM based transcription for any Indic scripts. There are a number of reasons behind this. The LSTM or HMM networks depend on character modeling and finding the characters from a word image. This often generates a stream of characters which includes many false detections and duplicates of similar characters which are usually compiled with a grammar or language based model. Most of the Indic scripts do not have such a strong model ready to be used. The biggest problem for such character modeling of scripts like Bangla comes from the diacritic or conjunct like attributes. An LSTM network sweeps horizontally, but a diacritic-character or character-character combination of Bangla can share vertical spaces with the components too as shown in Fig 1.8 and 1.9. The same is true for conjuncts as well. This is not only true for Abugida scripts, but also many others like Korean.

When characters or other symbols share vertical spaces when written, the whole combination has to be treated as one character if it is to be modeled with an LSTM or HMM based architecture. This leads to the problem of having a massive number of classes (considering combination of characters) to be modeled and thereby the requirement for dataset grows while the recognition speed and performance falls. Therefore, although theoretically sound, no notable offline transcription works have been reported for scripts like Bangla or Korean.

2.4 Available Datasets

Datasets are one of the most crucial components for training as well as benchmarking the statistics of handwriting recognition algorithms. Often, the development of any particular algorithm depends on the existence and availability of rich and useful datasets. The Center for Microprocessor Application for Training Education and Research in the Computer Science and Engineering Department of Jadavpur University in Kolkata has a repository (CMATERdb) of simple and compound Bangla characters, numerals, common words, etc., [78, 79, 80, 81, 82]. They have line and word level ground truth tagging, although it was inaccessible at the time of this writing. Bhattacharya et al. [83] presented a dataset (ISI db) of isolated basic and compound characters, numerals and vowel modifiers. This dataset is accessible by request. Biswas et al. [84] also presented a publicly available dataset (BanglaLekha-Isolated) which consists of isolated basic characters, numerals and a few high frequency conjuncts. Adak et al. presented a dataset called NewISIdb [67] which contains word images and covers many high frequency words and syllables. A summary of these datasets is presented in Table 2.1. Beyond these, several works

indicate the existence of other datasets, but none of them are publicly available for others to access.

Table 2.1: Existing public Bangla Handwriting Datasets (Numbers presented here are close estimates)

Attributes	**CMATER Dataset [78]**	**ISI Dataset [83]**	**BanglaLekha Isolated [84]**	**NewISIdb HwW & HwP [67]**
Basic Characters	15,000	30,000	98,000	Present*
Numbers	6,000	23,000	19,000	Present*
Characters with Vowel Diacritics	None	Present*	None	Present*
Consonant Conjuncts	42,000	Present*	47,000	Present*
Essay/Paragraph	150** pages	None	None	1,07,550 words
Ground Truth Metadata	Line and Script Level Information**	N. A.	N. A.	Word Level
Accessibility	Open	On Request	Open	On Request

*Exact numbers couldn't be found.

**Couldn't be accessed during the time of writing

CHAPTER 3

DESIGN OF THE OFFLINE HANDWRITING RECOGNIZER

3.1 Overview

The fundamental contribution of this dissertation is a design for an offline handwriting recognition method. As described in Chapter 2, this problem is still unsolved. There are partial solutions for some popular scripts and none for many others. Furthermore, there are no methods which can work script independently. Therefore, our most crucial contribution to this field is a method which we call "Character Spotting" that can be used to recognize any alphabetic script. This is a segmentation-free approach which is fast, robust and high-performing. Unlike word spotting algorithms and many other traditional approaches described in Chapter 2, this method neither depends on having prototype words in the document, nor is it restricted to a limited vocabulary. Rather this works for the entire script and for any script as long the script uses a limited number of characters. This technique was primarily designed to work for Bangla and was thoroughly tested on this script. Later it was shown how it can be adapted to work for other writing systems using Korean as an example.

One of the most limiting steps of developing an offline recognition system is

the dataset preparation. Although, collecting handwriting samples from many people with proper demographic sampling can be very difficult, the most costly process (in terms of manual labor and time) is actually preparing the dataset (such as ground truth labeling of characters, words, etc.) to make it suitable for machine learning. Here I present a method of "Autonomous Tagging" complementing the proposed offline character spotting recognition framework. This takes away most of the difficulties involved in dataset preparation. This too is an extremely flexible process and can work for any alphabetic script. The combination of the autonomous tagging and the character spotting recognition framework makes the development process of an offline recognition system for any writing system very easy and effective.

I also developed an offline handwriting dataset for Bangla named the "Boise State Bangla Handwriting Dataset". This is by far the richest dataset for Bangla and one of the richest for any Indic script. Extensive research was done to identify the most used conjuncts in Bangla and combine all of them in a short piece of text. This dataset efficiently captures most of the variations of the Bangla script. Most of the development and training for my experiments are conducted using this dataset. This dataset has been made publicly available, which can facilitate additional growth in the development of Bangla offline recognition.

Several other small experiments were done in conjunction with these major contributions. We developed an isolated basic character recognizer for Bangla to benchmark the Boise State Bangla Handwriting dataset with the other publicly available datasets for Bangla handwriting. This produced the highest recognition accuracy ever reported for Bangla basic characters. Another experiment was

conducted to compare the recognition performance between scanned and camera acquired data, where the rest of the framework and process remain the same. This experiment was made possible since the Boise State Bangla Handwriting dataset has most of its data digitized using both a flat-bed scanner and camera. Another experiment was done to see how the ground truth element position tagging accuracy affects the recognition performance. In other words, how much inaccuracy can the recognition algorithm tolerate in the ground truth tagging and still succeed in the recognition process. The outcome of this experiment not only proved the robustness of the proposed offline recognition design, also it motivated how the autonomous tagging method should be deployed.

This chapter describes the design, development and experiments in the following progression:

1. The design of the offline character spotting handwriting recognizer, along with how it works for Bangla and Korean, and why it should work for any alphabetic script.

2. Description of the underlying tools and technologies used to build this framework.

3. Description of the Boise State Bangla Handwriting dataset, the tools developed to process this dataset and the external datasets used for the experiments.

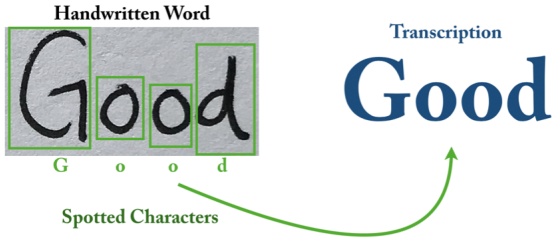
4. Experiment description to determine the effect of ground truth tagging accuracy over recognition performance and how the outcome result is used to develop the Autonomous Tagging framework.

3.2 The Proposed Offline Recognition System

3.2.1 Basic Idea of the Design

The proposed design for offline recognition works based on spotting or locating characters within a word. The core of this system is an object detection network which is trained to locate as well as identify characters in a given word image. This process is explained within the context of the Latin script in Fig 3.1 (a). As seen with this example word "Good", the object detection network attempts to find the letters from "A/a" through "Z/z" in the word. This is not a sequential process, the character spotting algorithm runs for all possible characters simultaneously. After it finds all the matches, we use the classes of the characters found (like one "G", two "o"s and one "d" in the example) and relative location information (the "G" is detected to the left of an "o") to obtain a transcription. This process is illustrated in Fig 3.1 (b).

(a) The characters are spotted in handwritten word image



(b) The transcription is obtained from the spotted characters

Figure 3.1: Basic idea of the presented offline character spotting recognition method: (a) shows an object detection network that attempts to find character matches in the word, and (b) shows the transcription formed using all the detected character class information.

This whole idea of character spotting is novel and not an incremental modification or combination of ideas of some existing approaches. Word spotting algorithms as described in Section 2.3 can be analogous since our method spots the characters in words instead of words in a page, but with a distinct difference between these two infrastructures. In our approach, we are recognizing the character class while spotting its location. Word spotting techniques just find the words based on a query without knowing the inner character composition and therefore cannot be used for transcribing. Also word spotting relies on a sample from that

document and is not robust against variations in handwriting. These methods are usually useful for indexing documents, finding keywords, doing primary level analysis, etc. On the other hand, since character spotting provides us both the classes of the characters and their spotted locations, it can easily be utilized to create a transcription.

Some other segmentation-free approaches that we discussed in Section 2.3 can detect as well as recognize words, but only from a limited vocabulary. Since the number of possible words in any script is enormous, it is not practical to build a detection model for all possible words by showing hundreds of samples for each of them during training. On the other hand, any alphabetic script comes with a limited number of character symbols, therefore it is possible to train a network for detecting each of them. Scripts like Latin, which has only 52 different symbols in its alphabet, are very easy to work with. This is one reason why we chose Latin to demonstrate the basic idea, although it was never used to test the framework. Things get complicated for scripts like Bangla or Korean where elements like diacritics and conjuncts come to play as explained in Section 1.5 and 1.7. In the following subsections, we describe how this framework was implemented for Bangla and Korean as well as why it should work for any alphabetic script.

3.2.2 Implementation for the Bangla Script [85]

As we showed in Section 1.5, because of the diacritics and conjuncts the Bangla script can effectively have more than 2000 different classes on which the object detection network needs to be trained. Almost all Indic scripts have this attribute. This is a problem since for training we require hundreds of samples for each pos-

sible class. To address this issue, we deployed two strategies:

1. We prepared two networks, one for the characters and one for the smaller symbols (like diacritics) which appear around the other characters keeping the base shape unchanged. The first one is named C-Net, abbreviated for Character Network and the later one is named D-Net for Diacritic Network. The lists of symbols that each of these networks are trained to detect are shown in Fig 3.4. On each word image, both of these object detection networks are applied sequentially. The C-Net first locates and identifies the characters from the list, and afterwards D-Net does the same with the diacritics. Therefore, even if we trained C-Net and D-Net from the same compound characters (a basic/conjunct character with a diacritic) from repeatedly showing the same part in different context, the C-Net only spots the characters and D-Net only the diacritics when tested on unseen words. This process is schematized in Fig 3.2.

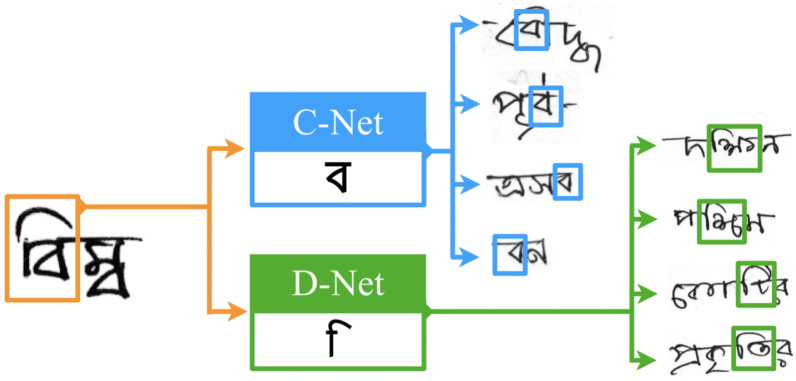


Figure 3.2: C-Net and D-Net work on the same image, but are trained to detect different symbol classes.

Once we obtain the detected classes from C-Net and D-Net along with their location information, a transcription is formed as shown in Fig 3.3. This two-

network-approach breaks the 2000 class problem into two smaller chunks and thereby the dataset requirement becomes much more manageable. This is applicable for any Abugida script, since all of these share similar attributes.

Another strong point of this approach is the individual networks are trained not only to spot the target characters, but also to ignore the surrounding characters/diacritics. This pattern of training makes the character spotting approach robust and insensitive to ground truth position tagging accuracy (demonstrated in Section 3.6.2) and allows an autonomous method of tagging to be implemented to avoid intensive manual labor during dataset preparation (demonstrated in Section 3.6.3).

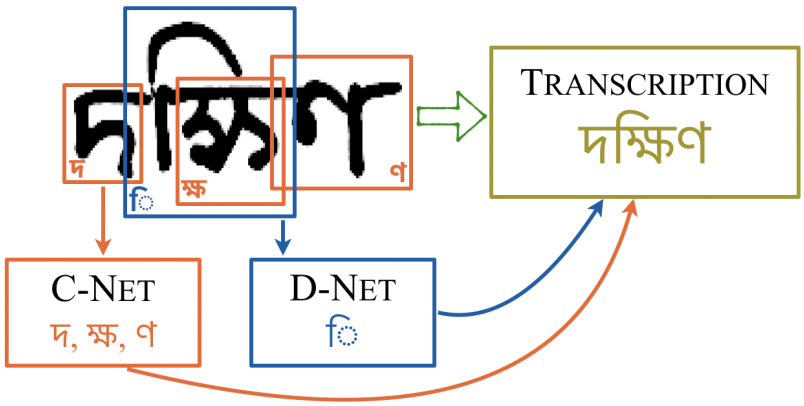


Figure 3.3: Detected symbols from C-Net and D-Net are merged to form a transcription.

2. The other strategy is to statistically reduce the number of unique appearances. In Bangla, like many other scripts, there are many character classes (mostly conjuncts) which are rarely used. To compensate for this, we analyzed a large volume of written documents to obtain a character set which covers more than

99% of the entire Bangla script. This approach reduced the class size by roughly 30% and made the whole process of dataset preparation as well as training much more convenient without sacrificing a substantial or even noticeable amount. The conjunct column in Fig 3.4 shows the reduced conjunct list we used for the development of our Bangla recognition framework.

C-Net Classes			**D-Net Classes**		
Basic	Conjucnt	Punctuation	Basic	Diacritic	Conjunct
অ, আ, ই, ঈ, উ, ঊ, এ, ঐ, ও, ঔ ক, খ, গ, ঘ, ঙ, চ, ছ, জ, ঝ, ট, ঠ, ড, ঢ, ণ, ত, থ, দ, ধ, ন, প, ফ, ব, ভ, ম, য, র, ল, শ, ষ, স, হ, ড়, ঢ়, য়, ং, ৎ	ন্দ, শ্চ, ঙ্গ, ধ্ব, গ্ধ, ষ্ট, ষ্ঠ, ষ্ম, প্ল, ন্ত, ন্ন, দ্ব, ক্ষ, ক্ষ, ত্ত, ট্ট, ষ্ণ, দ্ধ, ন্ড ক্ক, ক্ট, ক্ত, ক্ব, ক্র, ক্ল, গ্ন, গ্ব, গ্ম, গ্ল, ঘ্ন, ঙ্খ, ঙ্ঘ, চ্চ, চ্ছ, জ্জ, জ্ঞ, জ্ব, ঞ্চ, ঞ্ছ, ঞ্জ, ড্ড, ণ্ট, ণ্ঠ, ণ্ড, ণ্ঢ, থ্র, ত্ব, ত্ন, ত্ম, ত্র, থ্ব, দ্দ, দ্ভ, দ্ম, ন্ট, ন্ঠ, ন্ত, ন্ধ, ন্ব, ন্ম, ন্দ, প্ট, প্ত, প্ল, প্স, প্ত, ফ্র, ব্জ, ব্দ, ব্ধ, ব্ব, ব্ল, ম্ন, ম্ভ, ম্প, ম্ফ, ম্ব, ম্ম, ম্ল, ল্ক, ল্গ, ল্ট, ল্ড, ল্ব, ল্ল, ল্ফ, ল্ব, ল্ভ, ল্ম, শ্ছ, শ্ন, শ্ম, শ্ল, ষ্ক, ষ্প, ষ্ফ, ষ্ঠ, স্ক, স্খ, স্ত, স্থ, স্ন, স্প, স্ফ, স্ম, স্ল, স্ব, হ্ণ, হ্ন, হ্ম, হ্ল, ক্ষ্ণ, ক্ষ্ম, চ্ছ্ব, জ্জ্ব, ত্ত্ব, দ্ভ্র, ন্ত্ব, ন্দ্ব, ষ্ক্র, স্ক্র, স্ত্র, স্প্ল	‘, (কমা)’, ‘। (দাড়ি)’, ‘- (হাইফেন)’	ঁ	া, ি, ী, ু, ূ, ে, ৈ, ো, ৌ, ৃ	র্প, প্র (except for ক্র, ত্র), প্য, প্যা (প is used for base character only)

Figure 3.4: Class distribution for C-Net and D-Net training. Both networks are trained using the Boise State Bangla Handwriting dataset. The black colored characters are from the essay scripts and the blue colored characters are from the conjunct word documents as described in Section 3.4.

3.2.3 Implementation for the Korean Script

We used the same framework as for Bangla but with different strategies based on the unique attributes of the Korean script. As we described in Section 1.7, the Korean script does not have any diacritics, but has syllables with a two dimensional structure where the 28 Jamos can appear in different places in many different combinations. This also results in more than 2000 different possible combinational arrangements like Bangla and therefore the problem remains the same. However, the Jamos are written separately and are not inherently connected like Bangla via a "Matra" as can be seen from Fig 1.14 (b). Also, the Jamos do not change their shapes in their composite syllables like many Bangla characters do in their conjunct appearances. Therefore, rather than trying to recognize a Korean syllable as a whole, we tuned our object detection network to locate and identify the Jamos inside the syllable. Since the number of Jamos are much fewer than the number of possible syllabic combinations, our offline recognition process can be achieved using a much smaller dataset and shorter training than the traditional approaches. Therefore, while the basic idea remains the same as with the Bangla script, the execution was adapted to better treat the attributes of the Korean script.

Figure 3.5: List of detection classes of the Hangul script trained with K-Net.

Because there are no diacritics in Hangul, we prepared a single network K-Net (for Korean Network) to recognize the 33 character classes shown in Fig 3.5 from the compound syllables. These are the 28 basic characters or Jamos plus 5 more compound characters which are usually written in a connected form (classes 2, 5, 9, 11 and 14 in Fig 3.5). The rest of the architecture and tools are identical as for Bangla. The recognition process is the same regardless of how many Jamos are in the target syllable. Fig 3.6 shows an example syllable which is made of 4 Jamos. Rather than trying to recognize the syllable as a whole, the K-Net spots those 4 Jamos, then from those detections we re-construct the syllable. In this way, our object detection network only has to master spotting those 33 classes in Fig 3.5, rather than being trained to recognize more than 2000 possible syllable classes.

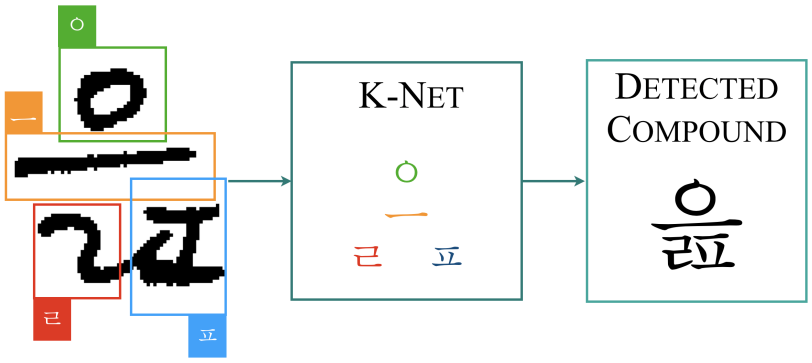


Figure 3.6: The Korean offline recognition process. This example shows a syllable made of 4 Jamos. Instead of recognizing the whole syllable, the K-Net only spots the Jamos. Later, using the detected classes and their corresponding locations, the compound syllable is constructed.

3.2.4 Implementation for any Alphabetic Script

The design of this offline recognition framework is such that we kept the script specific moderations on a separate domain keeping the core framework unchanged.

All the Abugida scripts (with vowel diacritics) can share the same framework with Bangla by using a sequential spotting of characters and diacritics with two detection networks. Pure alphabetic scripts like Latin, Arabic, etc. can be implemented just using a single detection network trained on their alphabets. We also showed how to handle scripts with compound syllables with the Korean script. Writing from right to left (like Arabic, Syriac) or top to bottom (Kulitan, Nushu) doesn't change the design core, but just the compilation order of transcription. Therefore any alphabetic or alphasyllabary scripts (which are constituted of a finite alphabet) can be implemented using this proposed approach.

3.3 Underlying Tools and Technology

The hierarchical architecture of this offline handwriting recognition framework is presented in Fig 3.7. The core of this framework is an object detection network which is trained to detect the characters or symbols. Afterwards, the detection results are compiled to form a transcription which requires script specific implementation. At the end, there is an optional post processing (such as spell checking) to further improve the result. This object detection network is developed from a pre-trained neural architecture using transfer learning. Then it is trained using an annotated dataset with standard data augmentation techniques. The green boxes in Fig 3.7 signify the portion of this framework which is script independent, and the red boxes highlight where script specific treatments are needed. Furthermore, the choice of the object detection network, data augmentation and transfer learning process are also flexible in this framework. This hierarchy is described in the following subsections.

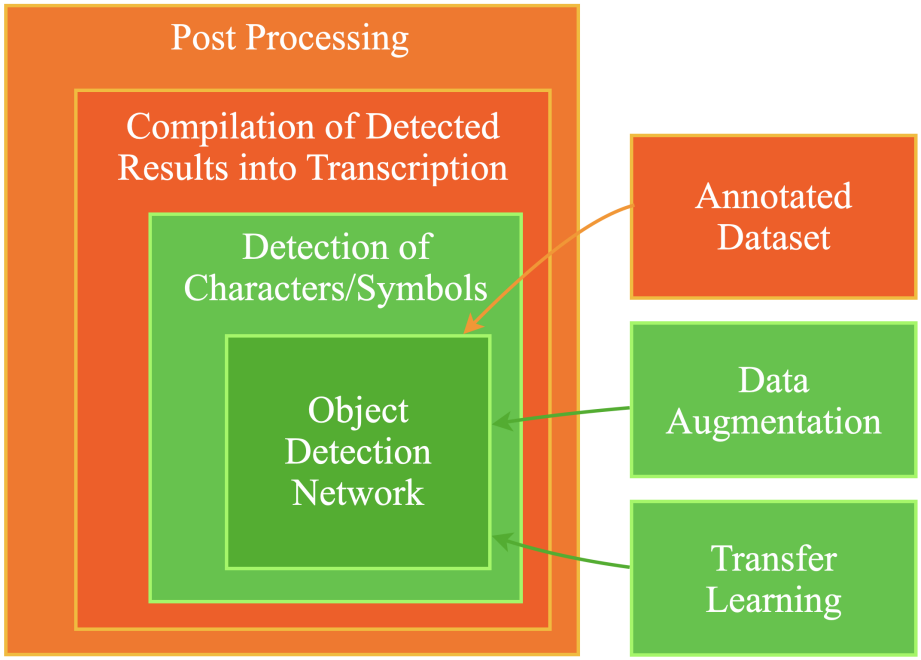


Figure 3.7: Hierarchical building block of the offline Handwriting Recognition Framework. Green highlights the sections which are script independent and orange highlights where the script specific details are implemented.

3.3.1 The Object Detection Network

As we mentioned, the choice of the core object detection network is flexible as it has been separated from the rest of the architecture. We chose a Faster RCNN architecture in our design. The Faster R-CNN, introduced by Ren et al. [86], is a faster variant and extension of R-CNN (Regions with CNN features) and Fast R-CNN. It provides almost real time object detection and has been used in many applications in machine learning including document image analysis. R-CNN and Fast R-CNN use a region proposal algorithm as a pre-processing step. In the case of Faster R-CNN, this issue is addressed by implementing the region

proposal mechanism using the CNN, hence making the region proposal a part of the training and prediction. The predicted region proposals are then reshaped using a RoI (Region of Interest) pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes. The cost of its fast detection speed comes from a slower training time. The applications related to handwriting recognition fields often require a faster recognition and very rarely require a quickly trained network. The Faster RCNN serves this requirement very well. Some other networks like YOLO (You Only Look Once) [87] or many HMM based approaches can also be a contender to be used in this framework.

3.3.2 Transfer Learning from VGG16 and Associated Parameters

No matter how successful it is, Deep Learning is notorious for being slow in training time. Even with dedicatedly designed powerful GPUs, this is much slower than most other conventional machine learning techniques. One way to mitigate this problem is referred to as transfer learning and this approach has been widely adopted. The idea is to use an already well trained network regardless of what it was originally purposed for, and re-tune the structure and network weights from that design. Not only does it make the process of network training much faster, in most cases it ends up producing a better design as a whole.

In our designs, we selected VGG-16, one of the most widely used pre-trained neural networks. It is sufficiently large to handle the large number of classes of Bangla, and not so large that it would be considered an overkill. Again, the choice of this network is flexible, any pre-trained network like VGG-32, RESNET, etc. can

be used with this approach. VGG-16 is a deep neural network introduced by the Visual Geometry Group (VGG) from the University of Oxford at the ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2014 [88, 89]. For our requirements, the series VGG-16 architecture was transformed into a DAG (Direct Acyclic Graph) structure to obtain a Faster R-CNN network, shown in Fig 3.8 with the pre-trained model weights. The building block of any CNN architecture is the convolution layer which is an application of a filter to an input that results in an activation. Repeated application of the same filter results in a map of activations (feature map) which indicates the locations and strength of a detected feature in an input. The ReLU works on a piecewise linear activation function which will output the input directly if it is positive or zero otherwise. The bounding boxes around potential objects in an image are handled with the Region Proposal Network (RPN) within the Faster R-CNN. A region proposal layer has two inputs - the classification scores produced by the RPN classification branch and the bounding box deltas produced by the RPN regression branch. A RoI (Region of Interest) max pooling network is used to output fixed size feature maps for all rectangular ROI within the input feature map in a Faster R-CNN architecture. The FC (Fully-Connected) layer takes input from the previous layer, calculates the class scores and outputs a one dimensional array of size equal to the number of classes. The SoftMax is an activation function that outputs a vector which represents the probability distributions of all potential outcomes. In our design, the features extracted from the ReLU5_3 (Rectified Linear Unit) layer was processed by a RoI (Region of Interest) pooling layer with 7×7 feature map output size replacing the last max pooling layer from the original VGG-16 architecture. All networks (C-Net, D-Net and K-Net) for our experiments were developed with this architecture.

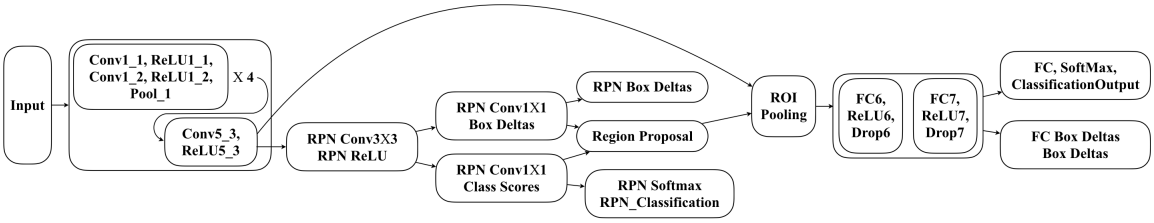


Figure 3.8: Layer graph of C-Net and D-Net, transformation of VGG-16 to a Faster R-CNN

Stochastic Gradient Descent with Momentum (SGDM) was used for training. The gradient descent algorithm updates the network weights and biases to minimize the loss function with small steps in the negative gradient direction of the loss

$$\theta_{l+1} = \theta_l - \alpha \nabla E(\theta_l) \tag{3.1}$$

where l is the iteration number, α is the learning rate, θ is the parameter vector, and $E(\theta)$ is the loss function. The gradient $\nabla E(\theta)$ is estimated with the whole training set. The standard gradient descent algorithm uses the whole dataset at once.

The stochastic gradient descent algorithm sometimes oscillates along the path of steepest descent towards the optimum. One way of reducing it is by adding a momentum term to the parameter update [90]. The stochastic gradient descent with momentum update becomes

$$\theta_{l+1} = \theta_l - \alpha \nabla E(\theta_l) + \gamma(\theta_l - \theta_{l-1}) \tag{3.2}$$

where γ determines the contribution of the previous gradient step to the current iteration. We used 0.9 as the value of momentum. The initial learning value was

set to 0.001 and the number of maximum epochs to 10. Negative trainings are done during the process by setting an overlap range with Intersection over Union (IoU) defined as:

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}\ or,\ \frac{area(A \cap B)}{area(A \cup B)} \tag{3.3}$$

where A and B are bounding boxes of the region proposal and actual value obtained from the ground truth file. Overlap ratios up-to 0.6 were used for negative training and higher values were considered positive. Lastly, the number of region proposals to randomly sample from each training image was set to 64. Increasing this number can obtain higher training accuracy at the costs of increased memory usage and a slower training process. C-Net, D-Net and K-Net, all were trained identically with these parameters, except for the difference in number of classes.

3.3.3 Data Augmentation

Data augmentation is a widely used technique to mitigate the problem of training with smaller datasets. This is a strategy that increases the volume and diversity of data without actually collecting new data. Techniques such as cropping, padding, stretching, adding skew, etc. are commonly used to generate augmented data. In most cases it improves the performance of the system while in rare cases it ends up making the system over-trained. At the beginning phase of creating the Boise State Bangla Handwriting dataset (when the volume of data was insufficient for proper neural training), we applied three basic but effective augmentation techniques:

1. Shearing along the X-axis (between -5° to 5°),

2. Rotation (between -5° to 5°), and

3. Scaling along the X-axis (between 50 - 150% of the original image width).

Augmented samples were made by randomly drawing levels for all three of these distortions simultaneously. For each word, we generated three additional images, thus quadrupling the training set size. Sample augmentation images are shown in Fig 3.9. Later as the dataset got bigger, we skipped this process.

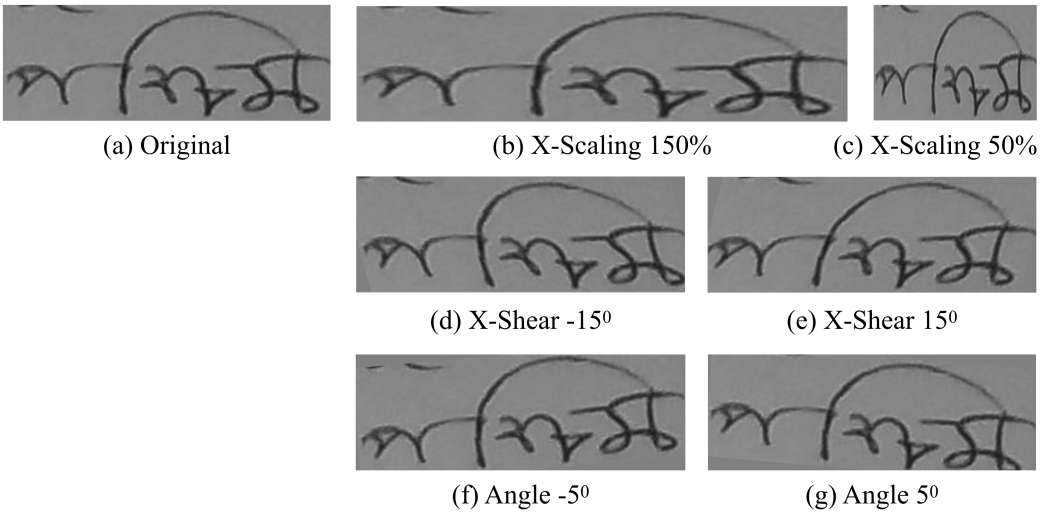


Figure 3.9: Data augmentation: (a) original, (b), (c) augmentation with only X-stretch of 150% and 50%, (d), (e) with only X-Shear of -15°and 15°and (f), (g) with only Rotation of -5°and 5°.

3.3.4 Compilation of Detected Results into a Transcription

As the networks (C-Net and D-Net) detect the classes and locations of the characters and diacritics present in a word, the information is compiled into a transcription or plain text representation of the word image. To achieve this properly, we stepped through a series of processing steps which are described in the

following subsections as well as demonstrated with an example in Fig 3.10. This step was done only for Bangla and not for Korean, since we used Korean only to demonstrate the script-flexible nature of our character spotting recognition framework. Note that the Korean transcription result can also be greatly improved using such sequence of processing and this script can also share some of these steps exactly as we did for Bangla.

(a) Original detection results overlay (orange for C-Net and blue for D-Net)

(b) Eliminate detection below confidence threshold (indicated by the Red box)

(c) Prioritizing detection overlaps (the red box encapsulated by the green box is eliminated)

(d) Empty space detected for possible missing element (indicated by green)

(e) Fixing the order of characters and diacritics (orange boxes come before blue boxes if overlapped)

(f) Spell correction (purple box replaces the missing element and the first diacritic)

Figure 3.10: Schematic of post processing for Bangla: (a) Original detections, (b) Eliminating detections below threshold, (c) Prioritizing detection overlaps, (d) Allowing empty spaces for possible detection miss, (e) Fixing order of characters and diacritics, (f) Spell correction.

Eliminate Detections below a Threshold

All detections are returned with a confidence value. Detections below a certain confidence threshold are considered to be unreliable and those were discarded from the transcription. The threshold is defined from the worst detection result of the networks. For C-Net and D-Net, the classes of ঊ (dirghô ū) and ঁ (ref) had the worst detection performances with *mAP*s of 0.66 and 0.79 respectively. The thresholds for these networks were decided from the *Precision-Recall* graph of these worst classes with values that maximize the *Recall*s. In information retrieval *Precision* is a measure of result relevancy, while *Recall* is a measure of how many truly relevant results are returned. The *Precision-Recall* curve shows the tradeoff between *Precision* and *Recall* for different thresholds. The values for the thresholds are 0.72 and 0.81 for C-Net and D-Net respectively, which is further discussed in Section 4.2 and shown in Fig 4.1.

Prioritizing Detection Overlaps

Many of the returned detections overlapped with each other. Some overlaps were expected such as a C-Net and D-Net overlap when there is a character/conjunct with a diacritic. Overlaps from a single network indicate multiple characters in one place. This phenomenon is expected for both Bangla and Korean (and most other scripts), because many characters/elements visually look like extensions of other characters/elements, and Bangla conjuncts often look like individual characters merged together. An analogous example will be the visual relations between Latin "c" and "e", or among "r", "n" and "m". A sample of C-Net detection overlaps

is demonstrated in Fig 3.11. We kept the largest bounding box discarding the smaller detected bounding boxes that it encapsulates or overlaps with regardless of their confidence score. The overlap was computed using the *IOU* Equation 3.3.

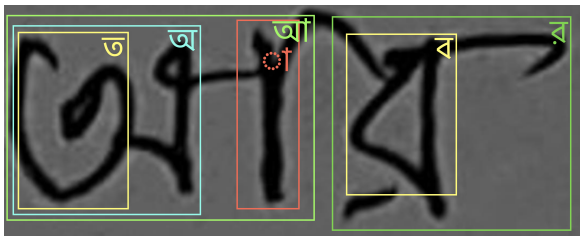


Figure 3.11: Sample of the detection overlap issue. Green boxes are the proper detection and the all other colored boxex are detected look-alike sub-characters, which are removed as errors

Allowing Empty Spaces for Possible Missed Detections

If two detection results have a large empty space in between them, the most likely reason is because a character was missed in the detection. These potential missed detection spots were identified and labeled with a blank character in order to properly assess the total number of characters in words. This helps the spell correction process described in Section 3.3.5. For Bangla, the following conditions are considered to be caused by a missed detection :

1. Two consecutive C-Net detections having a gap larger than 50% of the width of the first one is considered as a miss in detection, and thereby are filled in with a blank character.

2. There can't be two consecutive diacritics, so either one of them is a false positive, or there is a character missed in detection between them. For cases where

there is a big space gap between the two diacritics (set as 50% of the previous detection width), a blank character is inserted between them.

Using Script Knowledge to Compile the Transcription

Some script specific rules are applied at this stage to put the transcription in proper order, as well as to mitigate some of the detection errors. For Bangla the following properties are exploited to have a better transcription:

1. Some diacritics spatially start before the character and some afterwards, but with Unicode all the diacritics are encoded after their primary character regardless of where they visually appear. Therefore in all the overlapped results from C-Net and D-Net, the C-Net results were ordered first. There is one exception in the Unicode transcript with the conjunct class '◌্র' (Bangla conjunct 'ref'), which appears more like a diacritic and was processed with the D-Net. This actually appears before the associated character class in formation as well as Unicode encoding. Hence, for this case the D-Net result was placed prior to the C-Net one. The overlap was calculated in the same way as Eq. (3.11). With this exception, all the detected results were sorted with the x_{min} values of their bounding boxes.

2. A vowel can't have a vowel diacritic, therefore D-Net results overlapping with a vowel were eliminated.

3. A vowel can't form a conjunct, therefore if a C-Net result implied that it encapsulated a vowel with something else, based on confidence score one of them was eliminated.

4. As already mentioned, there can't be two consecutive diacritics. This was assessed by the space gap between the detected bounding boxes. For overlapped or tightly spaced detection, the diacritic with lower confidence was removed.

All these accommodations described above are different than a spell checker. A spell checker reduces the problem to a fixed vocabulary. These are mostly grammar-based rules and still allow an open vocabulary. Although the accommodations are script specific and therefore need to be modified for other scripts, most of the ideas remain same for all Abugida script.

3.3.5 Spell Checking

For Bangla, a basic spell checker was designed to improve the word recognition accuracy. This was particularly helpful at the beginning phase of this experiment where the dataset wasn't big enough for proper training. Later as the dataset grew, we removed this process from the framework. For this, a Bangla word library containing 450,000 words was used. With the corrections discussed in Section 3.3.4, the number of C-Net characters in a word was being estimated with 97.62% accuracy. Therefore, the insertion or removal of a character (from C-Net classes) was excluded from the edit distance calculation. With that we had:

$$Edit\ Distance = \frac{I_D + R_D + S_{C+D}}{No.\ of\ Elements\ Detected} \tag{3.4}$$

where I_D and R_D are the numbers of insertions and removals required strictly from the D-Net classes and S_{C+D} is the number of substitutions required from all classes. Words not included in the library were considered misspelled words and were replaced by the ones with the least edit distance from the detection.

3.3.6 Performance Metrics

The performance of the detection networks (C-Net, D-Net and K-Net) and the transcription unit were evaluated with *Precision*, *Recall*, mAP (mean Average Precision) values, $F1$ scores, WRA (Word Recognition Accuracy) and CRA (Character Recognition Accuracy). *Precision* and *Recall* are defined as:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}, \tag{3.5}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}. \tag{3.6}$$

mAP for a set of queries is the mean of the average *Precision* scores for each query. The $F1$ score (a.k.a F score) is the geometric mean of *Precision* and *Recall* as calculated by:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{3.7}$$

WRA is the ratio of the correctly transcribed words, among all test words which is the same as $1 - WER$ (Word Error Rate). CRA is $1 - CER$ (Character Error Rate), where CER measures the Levenshtein distance normalized by the length of the true word. The Levenshtein distance between two words is defined as the

minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other [91].

3.4 The Boise State Bangla Handwriting Dataset

3.4.1 Overview

One of the major difficulties with any machine learning problem is to prepare a dataset for training. Sometimes there are datasets developed by other researchers that are available for free and public use. Unfortunately, when we started this research there was no dataset available for offline Bangla handwriting which could be used with the proposed approach. There are some public Bangla datasets as discussed in Table 2.1, but none of them have character level ground truth position tags, which is a crucial metadata for character spotting. Therefore we created our own dataset which we call the "Boise State Bangla Handwriting Dataset" or the Boise State dataset [92]. This project was approved by the Institutional Review Board (IRB) at Boise State University. The dataset is freely available in the Boise State ScholarWorks at `https://doi.org/10.18122/saipl/1/boisestate`.

The Boise State Bangla Handwriting dataset contains both isolated characters and essay scripts all tagged at the character, word and line levels with associated ground truth metadata. This ground truth tagging is one of the key features of this dataset which other public Bangla datasets do not have. Participants from a variety of ages and professions contributed their handwriting samples for this dataset and their demographic metadata were also recorded. The text content was carefully

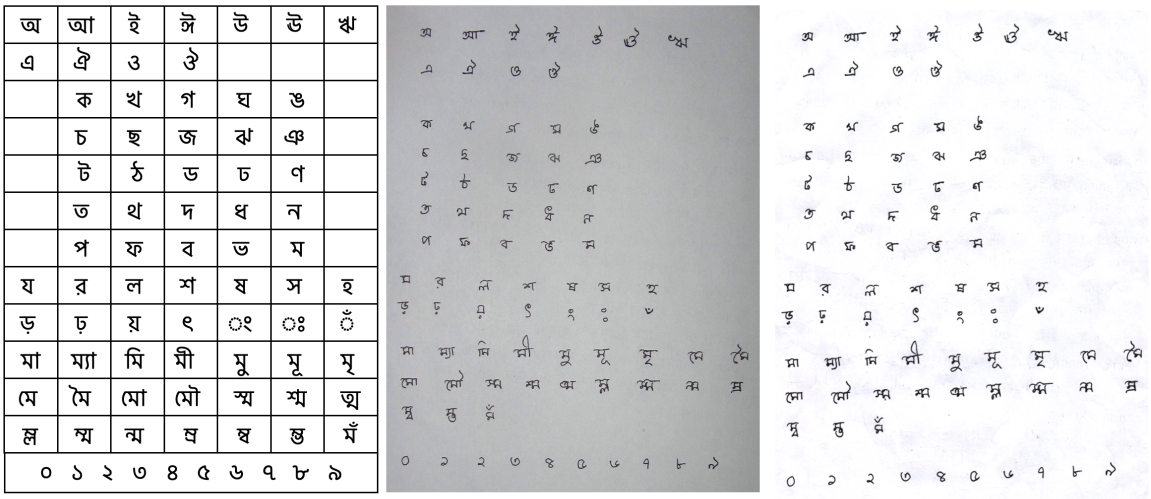
crafted so that it covers almost the entire Bangla script. Furthermore, the volume of this dataset is large enough to facilitate all kinds of approaches including Deep Learning. The details of the content and the tools developed to process this dataset are described in the following subsections.

3.4.2 Description of the Dataset Content

The Boise State Bangla Handwriting dataset has three kinds of content: a page of isolated components, an essay script and a number of conjunct heavy words. These are described next.

Isolated Component Document

The isolated components document of the Boise State dataset contains all 50 basic characters, 10 numbers, all 11 vowel diacritics with a consonant, 'ম' and 10 high frequency conjuncts. Fig 3.12 shows the content of this page in machine printed form as well as a camera-acquired and a scanned sample from the dataset. The purpose of this is to facilitate the isolated character recognition research which is important for applications as discussed in Section 2.2. There are 253 pages each written by a different writer.



অ	আ	ই	ঈ	উ	ঊ	ঋ
এ	ঐ	ও	ঔ			
	ক	খ	গ	ঘ	ঙ	
	চ	ছ	জ	ঝ	ঞ	
	ট	ঠ	ড	ঢ	ণ	
	ত	থ	দ	ধ	ন	
	প	ফ	ব	ভ	ম	
য	র	ল	শ	ষ	স	হ
ড়	ঢ়	য়	ৎ	ং	ঃ	ঁ
া	্যা	ি	ী	ু	ূ	ৃ
ে	ৈ	ো	ৌ	্ম	্ম	ত্ম
ল্ল	ম্ম	ন্ম	ন্ন	ষ্ব	ন্ত	মঁ
০	১	২	৩	৪	৫	৬	৭	৮	৯

(a) Machine printed version of the isolated components page

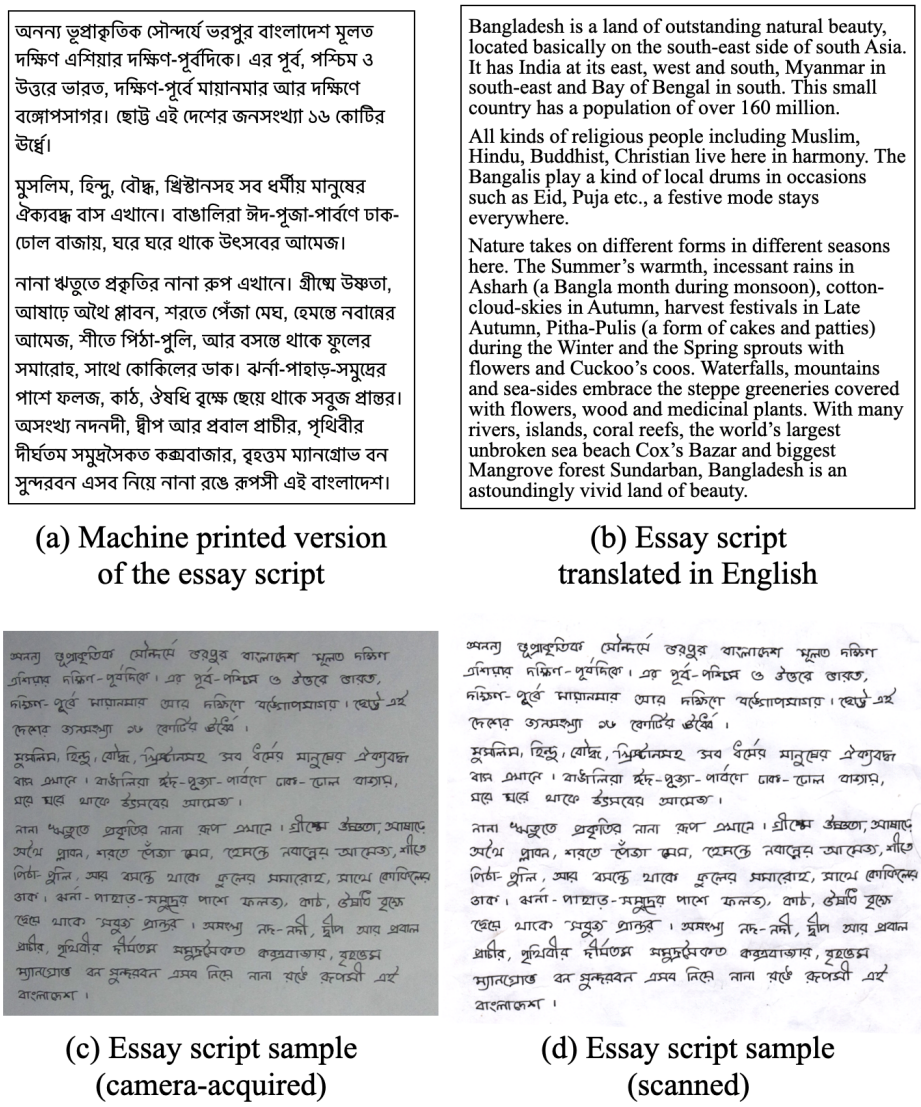
(b) Sample data (camera acquired)

(c) Sample data (Scanned)

Figure 3.12: (a) Machine printed version, (b) a camera-acquired sample and (c) a scanned sample of the isolated component document from the Boise State Bangla Handwriting dataset.

Essay Script Document

The second part of this dataset is an essay script, which is carefully scripted to contain all Bangla basic characters (except 'ঞ', which rarely appears in its basic form), all possible vowel diacritics and 32 high frequency conjuncts. Fig 3.13 shows the content of this script in machine printed form with English translation as well as a camera-acquired and a scanned sample from the dataset. The script contains a total of 104 words or 364 characters. The words used are mostly common and frequently used Bangla words. There are 253 pages of this script each written by the same writer group as the isolated component document.



(a) Machine printed version of the essay script

(b) Essay script translated in English

(c) Essay script sample (camera-acquired)

(d) Essay script sample (scanned)

Figure 3.13: (a) Machine printed version, (b) English translation, (c) a camera-acquired sample, and (d) a scanned sample of the essay script from the Boise State Bangla Handwriting dataset.

Conjunct Word Document

The third part of this dataset is a page of words containing the most frequently used conjuncts in the Bangla script. This was presented as a script format to the

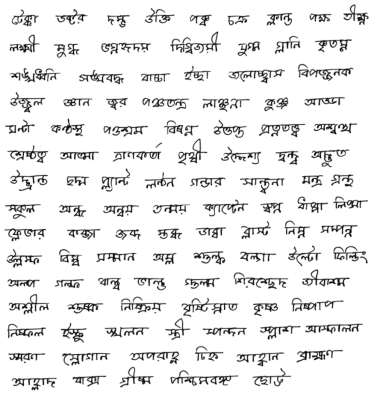
participants, not as a list of words. Fig 3.14 shows the content of this conjunct word document in machine printed form with English translation as well as a sample from the dataset. This document contains 128 high frequency conjuncts as shown in the conjunct column of Fig 3.4. These conjuncts were selected by surveying Bangla literature from books, web sites, magazines, etc. Our study shows these 128 conjuncts covers 99.7% of the complete Bangla script. Even from this list, some of the words were so rare that many participants had difficulties recognizing those words. The point of this is to ensure we cover the Bangla script as much as efficiently possible, keeping the dataset processing manageable. Beyond this, the remaining conjuncts not included in our dataset are mostly used to write names (or other nouns) which are translated from a different language. Along with the conjuncts, this page also contains some basic characters and diacritics. There are 70 samples from 70 different writers for this document.

টেক্কা ভক্টর দম্ভ উক্তি পক্ব চক্র ক্লান্ত পক্ষ তীক্ষ্ণ লক্ষ্মী মুগ্ধ ভগ্নহৃদয় দিগ্বিজয়ী যুগ্ম গ্লানি কৃতঘ্ন শঙ্খধ্বনি সঙ্ঘবদ্ধ বাচ্চা ইচ্ছা জলোচ্ছ্বাস বিপজ্জনক উজ্জ্বল জ্ঞান জ্বর পঞ্চতন্ত্র লাঞ্ছনা কুঞ্জ আড্ডা ঘণ্টা কণ্ঠস্থ পশুশ্রম বিষণ্ণ উত্তপ্ত প্রত্নতত্ত্ব অশ্বত্থ শ্রেষ্ঠত্ব আত্মা ত্রাণকর্তা পৃথ্বী উদ্দেশ্য দ্বন্দ্ব অদ্ভুত উদ্ভ্রান্ত ছদ্ম প্ল্যান্ট লণ্ঠন গন্ডার সান্ত্বনা মন্ত্র গ্রন্থ স্কুল অন্ধ অন্বয় তন্ময় ক্যাপ্টেন স্বপ্ন ধাপ্পা লিপ্সা ফ্লেভার কব্জা জব্দ স্তব্ধ ডাব্বা ব্লাষ্ট নিম্ন সম্পন্ন উল্লম্ফ বিম্ন সম্মান অল্প শুল্ক বল্গা উল্টো ফিল্ডিং অল্প গল্‌ফ বাল্ব ভাল্‌ভ গুল্ম শিরশ্ছেদ জীবাশ্ম অশ্লীল শুষ্ক নিষ্ক্রিয় বৃষ্টিস্নাত কৃষ্ণ নিষ্পাপ নিষ্ফল ইস্ক্রু স্খলন স্ত্রী স্পন্দন স্পৃলাশ আস্ফালন স্মরণ শ্লোগান অপরাহ্ণ চিহ্ন আহ্বান ব্রাহ্মণ আহ্লাদ বাহ্য গ্রীষ্ম পশ্চিমবঙ্গ ছোট্ট

(a) Machine printed version of the conjunct word document

Ace Doctor Pride Quote Digested Circle Tired Wing Sharp Lakshmi Enchanted Broken-hearted Conqueror Coupled Fatigue Ungrateful Sound-of-a-Conch United Kid Wish Spate Dangerous Bright Knowledge Fever Panchatantra Assault Bower Gossip Bell Memorized Pandemonium Depressed Heated Archeology Peepul Excellence Soul Savior Earth Purpose Conflict Strange Agitated Pseudo Plant Lantern Rhino Consolation Spell Book School Blind Relation Muse Captain Dream Hoax Yearn Flavor Hinges Seize Stunned Container Blast Low Completed Jump Reflection Honor Acid Tariff Rein Opposite Fielding Little Golf Bulb Valve Shrub Beheaded Fossil Obscene Dry Inert Rain-soaked Dark Innocent Ineffective Screw Loosening Wife Vibrancy Splash brandish Remember Slogan Afternoon Sign Invitation Brahman Delight Box Summer West-Bengal Small

(b) Conjunct word document translated in English



(c) Sample data of the conjunct word document

Figure 3.14: (a) Machine printed version, (b) English Translation, and (c) a sample of the conjunct word document from the Boise State Bangla Handwriting dataset.

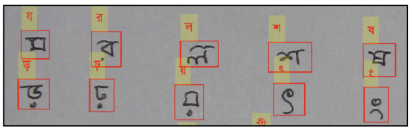
3.4.3 The Data Collection Process

The participation of the volunteers for this dataset was anonymous. The target content was provided in machine printed form as shown in Fig 3.12 (a), Fig 3.13 (a) and Fig 3.14 (a). Contributors copied the content on their own blank paper. The type of pen, pencil, paper and other writing environment were not specified to the participants in order to produce data with unconstrained handwriting. The conjunct word documents were digitized using only a flat-bed scanner. The rest of the documents were digitized in two ways - using different cellphone cameras and a flat-bed scanner as well. The cellphone acquired images were cropped and skew corrected and stored in "jpg" format. These images were captured using different cell-phones with different camera specifications. The resolutions of these images vary from 100 to 300 dpi with an average around 200 dpi. The scanned data were stored as "tif" in 300 dpi without any cropping or skew correction. No color alteration, resizing or filtering was done to any of these images. Digitizing data in multiple ways gives a natural form of data augmentation which is very useful for training. Furthermore, the data creates an opportunity for different kinds of experiments, one of which is described in Section 4.3.

3.4.4 Ground Truth Tag and Other Metadata

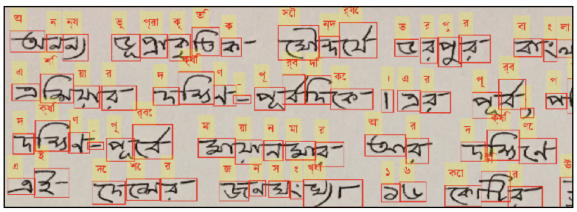
All data of the Boise State dataset were tagged with associated ground truth from all possible levels. The term "tagging" is used here to refer the process of finding bounding boxes that encapsulate the characters, words and lines from the document image, and storing the coordinate information along with the ground

truth character, word, line labels in a separate metadata file. The bounding box coordinate values are stored with x_{min}, y_{min}, *height* and *width* values. To keep this simple, we stored the metadata in a plain text (*.txt) file. A sample of the ground truth tag file with overlays on the data images for each kind of documents in the dataset is shown in Fig 3.15. We developed a set of special tools to achieve this as will be described in Section 3.4.6.



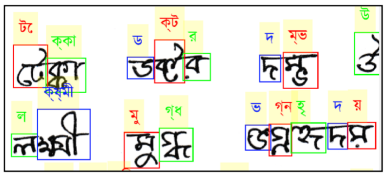
```
Char037 ষ - 115,1136,66,66
Char038 র - 279,1139,69,82
Char039 ল - 487,1161,88,60
Char040 শ - 692,1164,101,69
```

(a) Isolated component ground truth tagging and overlay



```
Line001 Word001 Char001 অ - 115,70,94,57
Line001 Word001 Char002 ন - 213,80,45,47
Line001 Word001 Char003 ন্য - 257,79,69,73
Line001 Word002 Char004 ভূ - 390,79,78,72
Line001 Word002 Char005 প্রা - 466,80,75,63
Line001 Word002 Char006 কৃ - 547,79,56,72
```

(b) Essay script ground truth tagging and overlay



```
Line001 Word001 Char001 টে - 437,374,65,81
Line001 Word001 Char002 ক্কা - 499,399,76,65
Line001 Word002 Char003 ড - 654,397,50,41
Line001 Word002 Char004 ক্ট - 706,364,57,82
Line001 Word002 Char005 র - 760,390,52,51
Line001 Word003 Char006 দ - 906,393,40,51
```

(c) Conjunct word document ground truth tagging and overlay

Figure 3.15: Samples of ground truth tag metadata of the (a) isolated component document, (b) essay script, and (c) conjunct word document from the Boise State Bangla Handwriting dataset. The left images show the tag overlay on the documents and the right images show the recorded metadata.

We also saved the basic demographic information of gender, age, profession and left/right handedness information of the writers along their writing samples for the isolated components and essay script documents of this dataset. This added

metadata opens up the possibility of future demographic based research. The demographic distribution of the acquired data is shown in Fig 3.16.

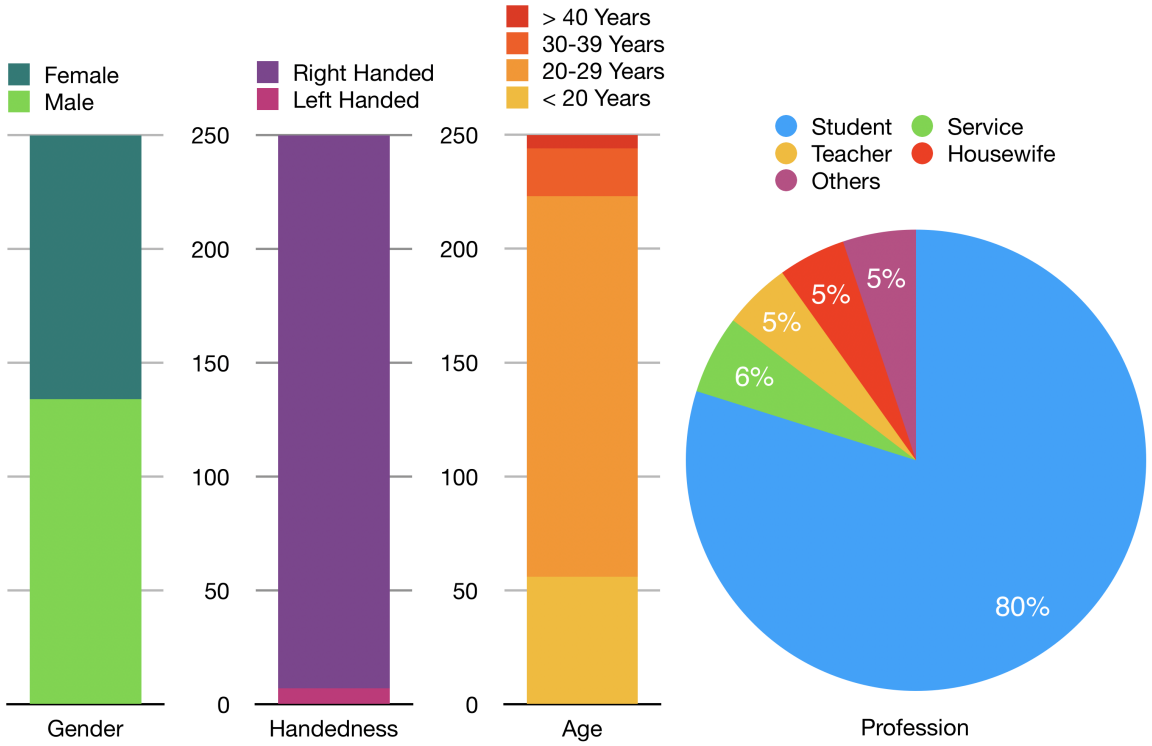


Figure 3.16: Demographic distribution of the writers for the Boise State Bangla Handwriting dataset. From left to right it shows the quantity and distribution of gender, right/left handedness, age and profession distribution of the participants.

3.4.5 Comparison with Other Public Datasets

The attributes of the Boise State Bangla Handwriting dataset together with the attributes of the other publicly available datasets from Table 2.1 are shown in Table 3.1. The ground truth tagging of the scripts at the character level is one of most the notable features of this dataset. No other public dataset for Bangla has character level ground truth information, which is crucial for our character

spotting framework.

Table 3.1: The Boise State Bangla Handwriting dataset compared with other publicly available datasets introduced in Table 2.1

Attributes	**CMATER Dataset [78]**	**ISI Dataset [83]**	**BanglaLekha Isolated [84]**	**NewISIdb HwW & HwP [67]**	**Boise State Bangla Handwriting Dataset [92]**
Isolated Basic Characters	15,000	30,000	98,000	Present*	12,650
Isolated Numbers	6,000	23,000	19,000	Present*	2,530
Isolated Characters with Vowel Diacritics	None	Present*	None	Present*	2,783
Isolated Consonant Conjuncts	42,000	Present*	47,000	Present*	2,530
Essay (# pages)	150**	None	None	107,550 words	323 Pages
Ground Truth Metadata	Line and Script Level Information**	N. A.	N. A.	Word Level	Character, Word, Line and Essay Level Information
Accessibility	Open	On Request	Open	On Request	Open

*Exact numbers couldn't be found.

**Couldn't be accessed during the time of writing

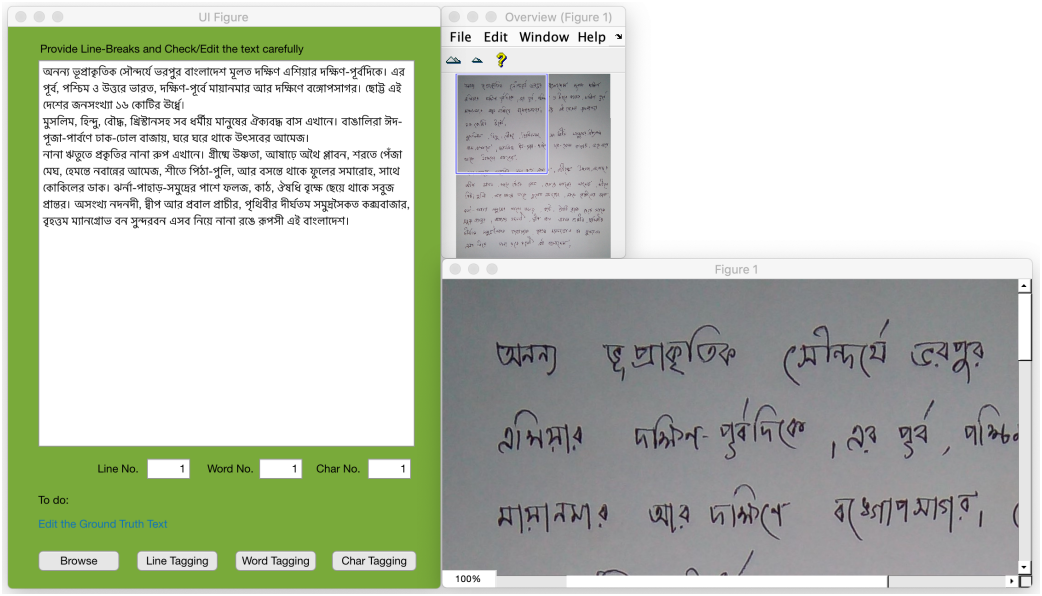
3.4.6 Tools Developed for Preparing the Dataset

In order to process the collected data, several applications were developed. These are quite versatile and can be reused for other similarly structured data regardless of what script it is. There are three tools developed in total as described in the following.

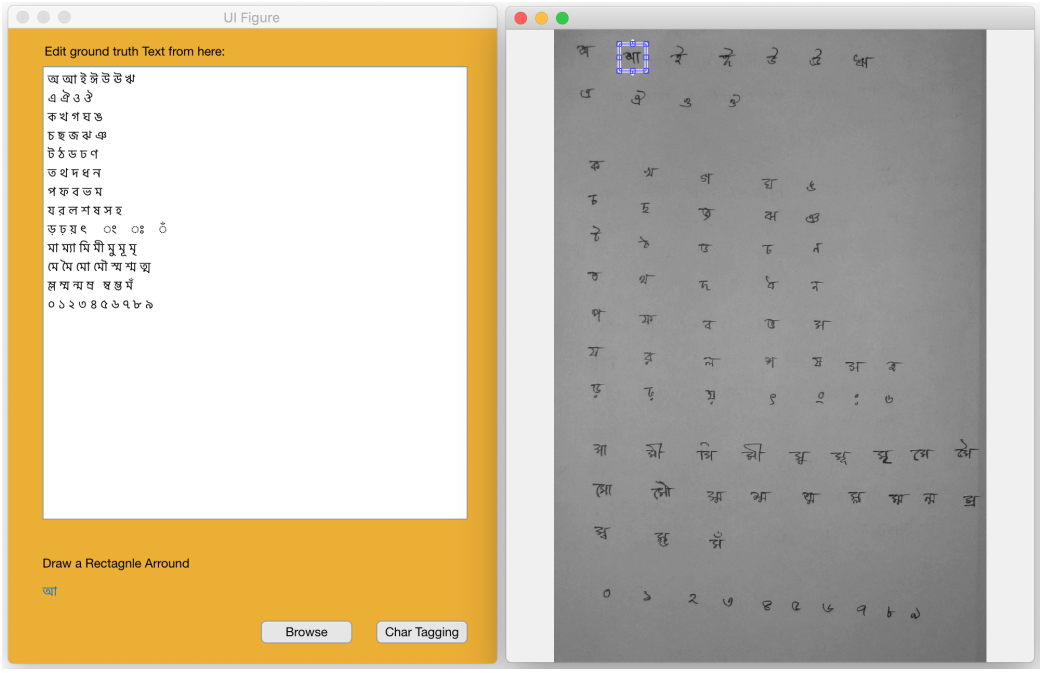
Tool for Data Tagging

Since we knew in advance what the text content would be on a particular document type of this dataset, a data tagging tool was developed to help draw the bounding boxes around each character/word/line and assign the text to that box. This tool first allows the user to edit the ground truth text content of the document

being processed, since in many cases the writers misspelled some words or totally missed a few. Afterwards the interface guides the user through the image to tag co-ordinates of all the characters in a rectangular way. It provides a control to zoom in or out as well as a navigation thumbnail to increase the convenience of finding the right spot on the page. During the operation it keeps track of how much text is already labeled and displays the next character of the specific word and line to be tagged. At the end, it creates a text file with all the line, word and character co-ordinate information. There were two versions of this app, one for the essay scripts and the other for the isolated character images. The working interfaces of these applications are shown in Fig 3.17. The samples of the resulting ground truth files are shown in Fig 3.15 (d, f).



(a) Essay Page Ground Truth Tagging Interface



(b) Isolated Components Document Ground Truth Tagging Interface

Figure 3.17: Working interfaces for the ground truth tagging application of (a) the essay script and (b) the isolated components document from the Boise State dataset.

Tool for Tag Verification

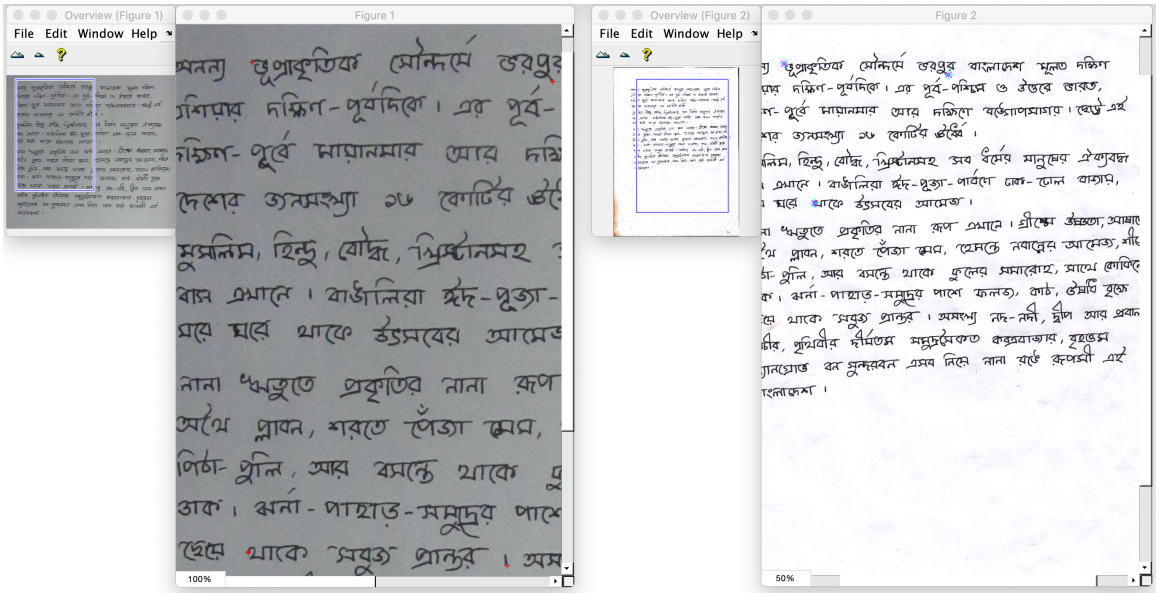
Another application was developed to verify the ground truth file with its corresponding image to scrutinize the content for mistakes. This tool can detect automatically whether it's an essay script or an isolated character page and changes its behavior accordingly. It receives the text file and the image as inputs and displays the image with overlay of the co-ordinates. Multiple color schemes were used for better visualization. A sample case is illustrated in Fig 3.18.

Figure 3.18: The tag verification application interface.

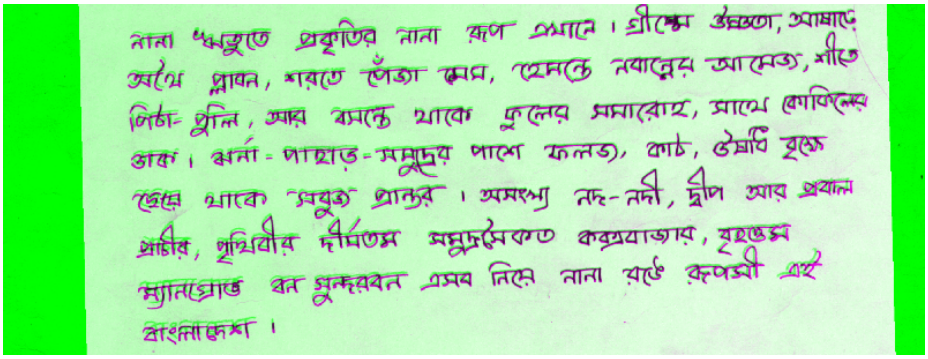
Tool for Tag Transfer between different Acquisition Sources

As mentioned earlier, all the data (except for the conjunct word pages) were digitized using cell phone cameras and also a flat bed scanner. In order to minimize the effort required for tagging the same data already tagged in an image

from a different acquisition source, this application was developed to transfer the tagging information from one image to the other by applying a geometric projective transformation. This tool displays the two images and asks the user to highlight a few pairs of corresponding points. The minimum number of points it requires is 4, but usually more are required to produce a better transfer operation. Using these control point pairs it aligns the pages and updates the co-ordinates by warping. Then it displays the result with a highlighted image, where the user can decide if he/she wants to add or modify any points. Once the user confirms that he/she is satisfied with the result, it stores the transformed co-ordinates in the same format as the original. Zooming and navigation interfaces were provided as well as a multicolor scheme was used for fast and easy operation. Also, a feature was added with which the user can set the horizontal or vertical offsets for shifting large group of co-ordinates. This application drastically reduced the labor and time of manually tagging the same image already acquired from a different source. A portion of the working interface of this application is shown in Fig 3.19.



(a) Working interface for transferring tag data



(b) Display of fusion of the images to verify the transfer

Figure 3.19: Tag transfer application (a) the working interface (b) Display of an overlay of the images to verify the operation.

3.4.7 Benchmarking the Boise State Dataset with an Isolated Character Recognizer

An isolated character recognizer was developed with a conventional machine learning approach to benchmark the Boise State Bangla Handwriting Dataset against three other publicly available datasets described in Section 3.5.1 [93]. The isolated characters (mostly in alphanumeric format) frequently appear in numerous places - such as document identifiers, forms, postal headers, house addresses, encrypted codes with confidential letters, handwritten flyers, posters, notices, banners, invitation cards, bank checks, tickets etc. in Bangladesh and a portion of India. Therefore, this research has its own potential to contribute to many common tasks such as machine sorting, task automation, etc. We used an SVM classifier based on a cubic kernel with extracted features based on zonal pixel counts, structural strokes and grid points with U-SURF descriptors modeled with bag of features. The details of this process are described next.

Pre-Processing

Before the features were extracted, all the sample images were preprocessed. First, a 2D Gaussian smoothing filter with standard deviation of 0.3 was applied. Then the color and grayscale images were converted to binary images using a threshold obtained using Otsu's method. The data from the BanglaLekha dataset [84] were originally binarized, therefore they were used without these two steps. Afterwards, area filtering was done to remove isolated small objects with an area less than 80 square pixels. The images were cropped to leave one blank or back-

ground pixel row and column on each edge. At the last stage, these were resized into a fixed height of 128 pixels, with a variable width to preserve the original aspect ratio.

Feature Extraction

Three categories of features were used for recognition. These are referred to as Zonal, Pattern and Gradient features.

For the **Zonal Features** the character images were split into equal 8×8 zones. From the binary images, where '1' represents the dark or object pixel and '0' represents the white or background, the features are computed as

$$R_{ij} = \frac{Sum\ of\ all\ Pixels}{Area\ of\ the\ block} \qquad i/j = 1, 2, ..., 8. \tag{3.8}$$

This creates a 64-bit vector mapping of different zonal footprints of the characters. This approach with different zone dimensions was also used by Bhattacharya et al. [83] to recognize basic Bangla characters.

For the **Pattern Features** processing was done to extract stroke directions for the samples. At the first stage, using a morphological operation the interior pixels of the object were removed leaving a thin outline of the connected border pixels [94]. All the connected objects in a column are replaced by only one center element of that object. The top- and left-most pixel is counted as the first key point and a column-wise search operation traces the stroke edge. The character boundary contour is followed. Points where the direction transitions from left to right, right to left, up to down or down to up are considered as other key points. If the

boundary leads to a dead-end or a branch with a length less than 1/4 of the image height, the trail is removed. If two key points are very close (measured by a Euclidean distance less than 1/25 of the image height), the later one from (the tracking direction) is removed. After these stages, a minimal clean outlined version of the characters are found with the highlighted key points. Next, the angles of the straight lines connecting adjacent key points are calculated. These angles are quantized in 45° intervals (8 compass directions). Then the Euclidean distances between interconnected adjacent key points are computed. Adjacent lines having the same angle (after quantization) were merged and any connection less than a threshold (1/5 of the image height) was ignored. The resulting connections, which represent the stroke direction pattern feature of the sample characters, are represented using a numeral string.

Fig 3.20 shows the various stages for obtaining these stroke direction pattern features and a few samples of the strings obtained for particular classes. Afterwards, a histogram of the unit elements and bigrams of these representative strings were taken as features. A total of 64 features were obtained in this process. These were normalized before being used as the second portion of the feature vector.

Lastly, the length of all the combined strokes of vertical lines (‘2’s and ‘8’s), positive slants (‘3’s and ‘7’s) and negative slants (‘1’s and ‘9’s) are calculated, normalized and used as a 3 dimensional feature vector along with the pattern features. The horizontal strokes (‘4’s and ‘6’s) are ignored in this case, because the majority of these strokes belong to the Matra. The use of the Matra varies significantly depending on handwriting style and never causes a misclassification of the basic characters. Some conjuncts and numbers have some conflicting attributes with the

basic characters based solely upon the presence of this Matra, but since this work is only for the basic characters, the horizontal stroke contributions are totally ignored in this stage.

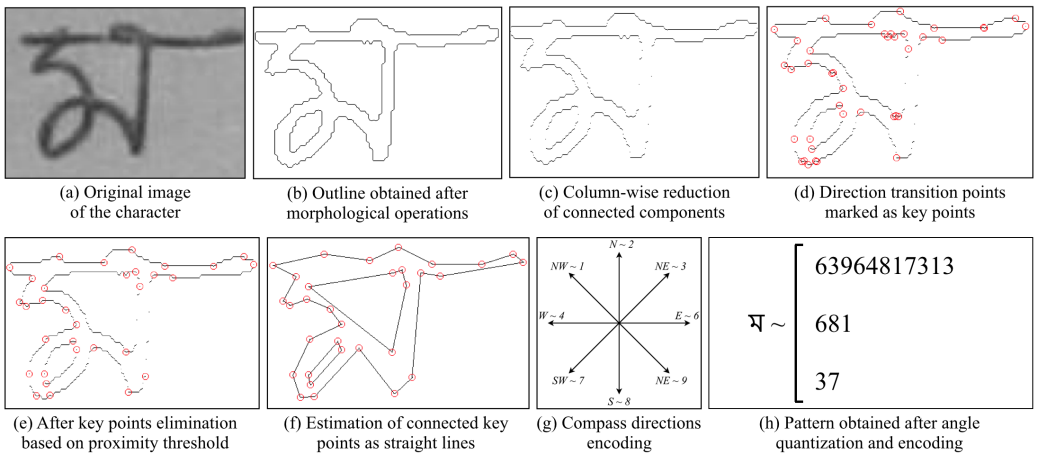


Figure 3.20: (a) to (h) illustrate the process of obtaining the pattern features for a sample character

For the **Gradient Features**, a uniform 8×8 grid was created on the sample. Upright Speed Up Robust Features (U-SURF) [95] were extracted from the intersection of the grid lines. U-SURF is a high performing scale invariant interest point detector and descriptor, although here only the descriptor was used to obtain the feature vector. Patch sizes for multi-scale extraction were selected as blocks of 32, 64, 96 and 128 square-pixels around the center. The upright version of SURF is not invariant to image rotation which makes it computationally faster and better suited for the cases where the camera remains more or less horizontal. The feature descriptor is based on the sum of the Haar wavelet response around the point of interest. The responses are then weighted by a Gaussian function with the interest point at its center and addressed as points in a 2D space with abscissa and ordinate

as the horizontal and vertical responses. The summation of the horizontal and vertical responses forms a local orientation vector. To describe the point, a square region around that point is extracted, divided into 4×4 square sub-regions, and each the Haar wavelet responses is approximated at 5×5 regularly spaced sample points. 80% of the strongest features from each sample were kept and fed into a bag of features model.

As we discussed in Section 2.3, Bag of Features representations have become very popular for their simplicity and great performance, and have been used in handwriting recognition quite frequently [58, 54, 96]. The basic idea of this approach is to take a set of local image patches (in this case U-SURF descriptors) and convert the vector-represented patches into codewords, which can be considered as representative of several similar patches. The collection of all the codewords is referred to as a codebook. This terminology is analogous to the concept of words and a dictionary from a document corpus. Afterwards, using K-means clustering, a 500 word visual vocabulary was prepared. Each patch in an image was mapped to a certain codeword and the image was represented by the histogram of the codewords.

From the zonal, pattern and gradient features a combined 631 dimension feature vector is prepared and fed into the classifier. Fig 3.21 shows the overview of all these feature points extraction from the pre-processed image of a sample character.

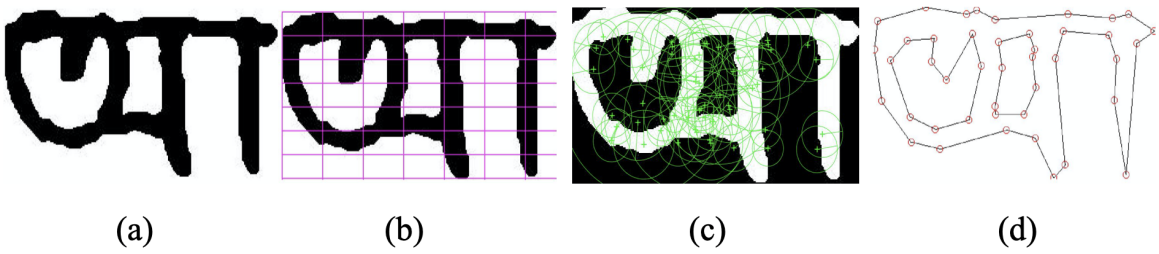


Figure 3.21: The extraction process of all feature points. The (a) pre-processed image, (b) zonal features, (c) pattern features, and (d) gradient features.

Classifier

A Support Vector Machine (SVM) was used on the feature vector obtained from the character samples. An SVM is designed for two-class pattern recognition problems. Multi-class SVMs are realized by combining several two-class SVMs. Here, a OVO (One Versus One) multi-class classifier was used as it offers better accuracy. The classifier was tuned with a cubic kernel. A cubic kernel is defined as

$$K(x, y) = (x^T y + c)^3 \tag{3.9}$$

where x and y are the feature vectors in the input space. The higher degree polynomial allows a more flexible decision boundary. Although non-linear SVMs are expensive to train, they performed significantly better than the linear SVMs in this case. All the features are normalized prior to the classifier input.

3.5 External Datasets used for the Experiments

3.5.1 External Bangla Datasets used for Isolated Character Recognition

To benchmark the character set from the Boise State Bangla Handwriting dataset (described in Sections 3.4.7 and 4.1) we used three other publicly available datasets:

1. **CMATERdb 3.1.2**, developed by the Center for Microprocessor Application for Training Education and Research (CMATER) in the Computer Science and Engineering Department of Jadavpur University in Kolkata,

2. **ISI handwritten basic Bangla characters**, developed at the Indian Statistical Institute (ISI), Kolkata, India and

3. **BanglaLekha-Isolated database**, developed by the Information and Communication Technology (ICT) division, Bangladesh.

Further details of these datasets were presented in Section 2.4 and Table 2.1. Only the isolated basic characters from these datasets were used to benchmark the Boise State isolated character dataset.

3.5.2 External Bangla Datasets used for Transcription Evaluation

One of the best ways to test the strength and robustness of a recognition system is to test with a dataset which is completely different than the one with which it was originally trained. With this aim we used three other datasets to test our Bangla recognition framework. These are all different from the native Boise State

Bangla Handwriting dataset as they are:

- Developed in Kolkata, India. The Boise State Bangla Handwriting dataset, with which our detection networks were trained, is made of contributions from people all from Bangladesh. Although the script is the same, there are differences in handwriting between these two countries. This might not be instantly apparent to most people, but could be substantial in machine learning.
- Collected with different acquisition processes, such as different scanner, settings, pre-processing, paper type, etc.
- Different in context. They all contain many different words or compositions which our system had never seen during training.
- Written by entirely different set of writers with different demographic distribution. Therefore a test with these datasets can ensure our offline recognition framework does not have any bias to a particular type of demographic.

In a sense, the success with these datasets actually reflects the true potential of our presented offline recognition system in the real life practice. The descriptions of these datasets are briefly discussed in the following and a sample from each of them are shown in Fig 3.22.

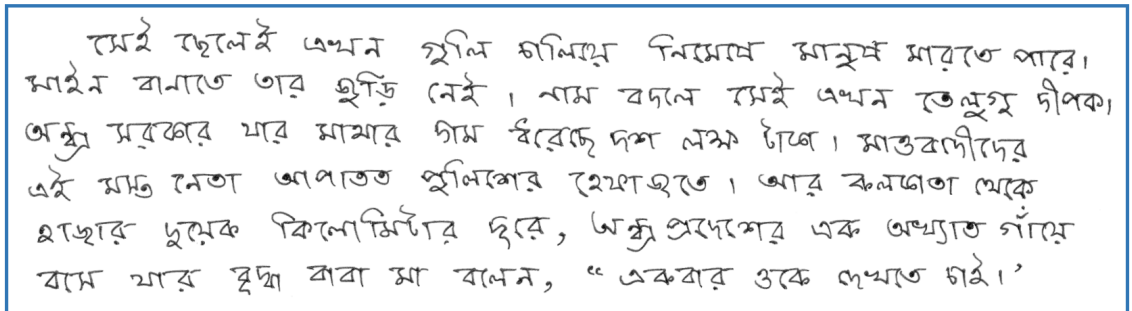
CMATERdb 1.1.1 [79, 78]

The CMATERdb 1.1.1 is one of the oldest and most used datasets for offline Bangla handwriting research. CMATER stands for Center for Microprocessor Ap-

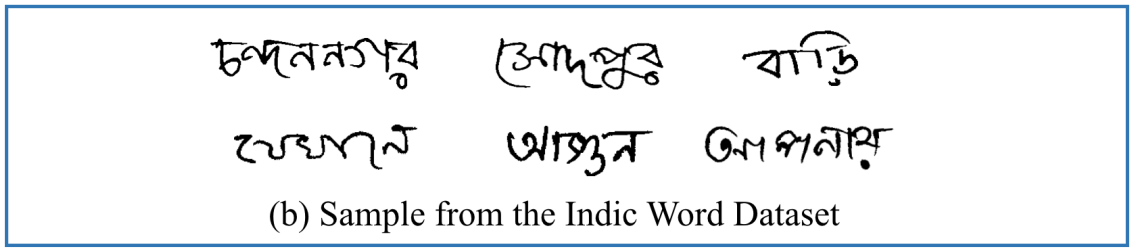
plication for Training Education and Research, developed at the Computer Science and Engineering Department, Jadavpur University, Kolkata, India. This contains 100 pages of unconstrained handwritten documents scanned and stored in 24-bit BMP format. Although, there are no transcriptions publicly available for this dataset, the CMATER group provided us with segmented word coordinates for a few of these documents.

Indic Word Dataset [97]

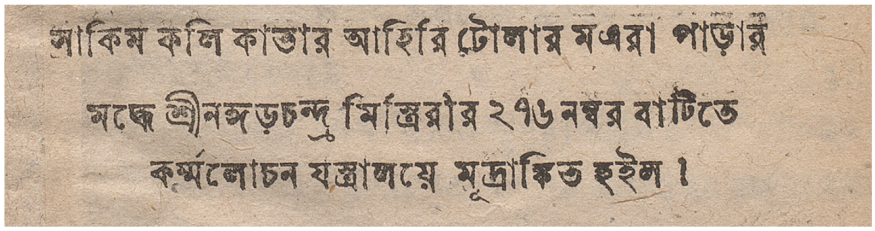
This is a relatively new dataset compared to the CMATERdb 1.1.1. The content of this dataset is segmented Bangla word images, not pages. This also comes with a transcription for each word, therefore we could use our framework on this dataset almost instantly. Instead of standard Unicode they used Latin counter-forms for Bangla characters, which we needed to convert before using this dataset. Right now this dataset is not publicly available and we are grateful to Pradeep Kumar to sharing this with us for testing. This contains 17,091 handwritten word samples with 1,736 unique words. The words are collected from 60 handwritten document images by writers of various professions. We tested our system with the test set from this dataset which contains 3,856 words.



(a) Sample from the CMATERdb 1.1.1



(b) Sample from the Indic Word Dataset



(c) Sample from the REID2019 Early Indian Printed Documents

Figure 3.22: Samples from the other datasets on which our framework was tested.

REID2019: Early Indian Printed Documents [98]

The REID2019 dataset is not an offline handwriting dataset, rather it is a set of scanned historical printed documents. Regardless of this major difference in document type, we still wanted to test our system's strength with this. Historical documents are often very tricky to deal with because over the ages they suffer

from tears and are worn, which results in many different distortions. Additionally, the content used here is almost archaic and many words are rarely used today. Since our approach doesn't depend on any restricted vocabulary, this was very interesting to see how it handles worn out archaic printed documents. Originally this dataset was used for a competition at the ICDAR (International Conference on Document Analysis and Recognition) conference. This has many kinds of meta-data that our system depends on, thereby the testing process was relatively easy. The transcription and word level coordinate information was provided with this dataset, therefore this too was instantly applicable to our approach.

3.5.3 Korean Dataset used for Syllable Recognition

We used only one dataset to test the functionality of our method on the Korean script. This is called the PE92 dataset [51], which is one of the most popular Korean handwriting datasets. This was collected by POSTECH, funded by ETRI (`www.etri.re.kr`) in 1992. It is a large dataset containing images of 2350 classes of syllables (not Jamos) and about 100 instances of each class, roughly 88% and 12% of them are labelled as training and test sets respectively. The ground truth is available at the syllable level, but not at the basic character or Jamo level which we needed. We used both manual and autonomous tagging on a small subset of this dataset, details are provided in Section 4.4.2. This was a small scale experiment just to demonstrate how the recognition framework is scalable to fit other scripts.

3.6 Autonomous Tagging

3.6.1 Background and Motivation

One of the most time consuming parts of preparing a classifier is collecting and annotating datasets. In this section, we present a simple yet extremely powerful idea which can drastically reduce the manual effort and time required to prepare an offline handwriting dataset that can be used with our character spotting method. The term "Autonomous Tagging" is used here to refer to an automated process of drawing bounding boxes at the character level on handwritten word images. Like the character spotting algorithm, the autonomous tagging technique is also flexible and can be used for any alphabetic writing system. Here we demonstrated the process with Bangla and Korean using the Boise State Bangla Handwriting dataset and the PE92 Hangul dataset.

Preparing a dataset is a major problem in the field of offline handwriting recognition. Not only is it tedious, the manual process is susceptible to human errors, which can cost more than the approach itself in terms of system performance. As we described in Section 3.4, the Boise State Bangla Handwriting dataset contains more than 323 pages of handwritten Bangla script with approximately 104 words or 364 characters per page. In order to develop and test our character spotting framework, all the words from this dataset needed to be tagged with their associated ground truth at the character level. This process involved roughly 600 hours of work, equivalent to a full-time job for almost 4 months - all for just one script. This time estimate includes using tools to facilitate the process and excludes

the time required for the tool development or data acquisition. Furthermore, it is a fatiguing process to do the tagging without making errors, which are really difficult to spot and fix later even with the tools described in Section 3.4.6. All these issues make this process so costly that there are a limited quantity of such datasets publicly available for open research. For example, none of the public Bangla datasets has character level location information and most of them do not even have word level ground truth tags as we presented in Table 2.1.

The process presented here is not intended to segment the characters, although some under-segmentation approaches may be used with our idea of autonomous tagging. A character segmentation algorithm requires much more precision and is usually combined with an isolated character/element recognition system to obtain a transcription. As we discussed in Section 2.2, segmentation-based approaches have not been very successful on handwritten text, and thereby are often discouraged in current research. An under-segmentation process attempts to enclose the target character in a large bounding box prioritizing the correct capture of the entire character more than the possibility of including the adjacent elements. Segmentation is a heavily script dependent process and is very tough to achieve correctly, especially with cursive and connected scripts. The autonomous tagging approach presented here is not intended to, nor it is precise enough to pre-segment the characters to be used later by an isolated character recognizer, rather it estimates the character locations with a loosely fit bounding box, which can later be used with a detection based transcription module like in the presented recognition framework. This type of estimation work has been attempted before, but mostly for printed words like Xu and Nagy's prototype [99].

With all the modern modeling techniques and tools, many researchers ask whether in order do unconstrained handwriting recognition, we need a dataset prepared with character level ground truth at all. This can be answered better by the fact that for Bangla prior to the existence of the Boise State dataset, there were no vocabulary free works or approaches reported which could dependably recognize unconstrained handwriting. As we discussed in Section 2.3, there are some works which recognize words as a whole unit [64, 65, 66], but they only work on a heavily restricted vocabulary and are useful only in specific applications such as postal automation. So no matter how difficult the preparation process is, it is actually very crucial to have such detailed datasets for handwriting research development. This is true not just for Bangla or Korean, but for most scripts. Many approaches, especially with object detection algorithms, rely on datasets with detailed annotated ground truth. Here, we present an approach for obtaining this kind of annotation with a dataset, but without the need for intense manual labor. Although the idea can be applied to almost any writing system, the process is not fully generic and has to be adapted to the attributes of each script. We tested this process with segmentation-free offline Bangla and Korean handwriting recognition, our character spotting framework, and compared the performance with the approach based on accurate manual tagging.

3.6.2 Effect of Tag Variance

Most datasets with ground truth tags are concerned with getting the tightest and most accurate bounding boxes for each script element and connecting a label to it [100]. The first study we describe consists of expanding the boundaries of

the precise manual ground truth tags or bounding boxes to observe how critical the precision is to the recognition method. Each character from the Boise State dataset was labeled using a rectangular box. As shown in Fig 3.23, we increased (or decreased) the width of these bounding boxes by an amount of -10%, 10%, 20%, 30% and 40% while keeping the height at the initial level (which is generally the word height) and recorded the detection performance in terms of mAP and $F1$ scores.

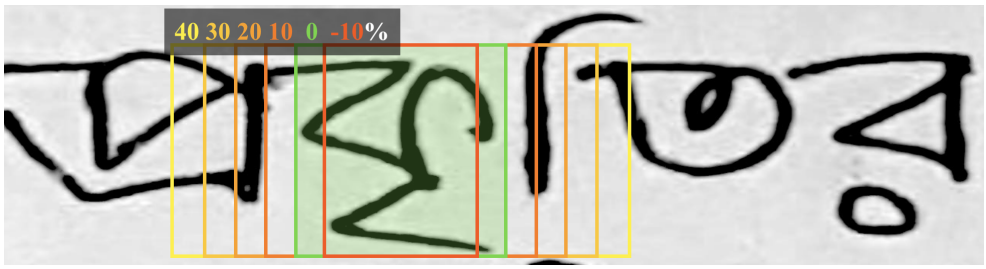


Figure 3.23: Bounding box widths were varied from the green box indicating the accurate location to -10%, +10%, +20% and +30% as shown by boxes of oranges and yellows.

The impact of changing bounding box widths on the recognition performance was measured with mAP, $F1$ score , CRA and WRA. These parameters are defined in Section 3.3.6. All values are converted to percent scale. The results are presented in Table 3.2. Fig 3.24 shows a plot of all these performance parameters versus the tag variations. As seen, while there is a decrease in performance as inaccuracies are introduced in the tag boxes' locations, the amounts are small. The system works with extended boundaries very well, but not when it is shrunk, since in many cases defining attributes of the characters/diacritics get cut off from the edges. But when extended, the performance degradation with the introduced

inaccuracies of tagging is minimal even at the 40% level. We used this observation as a foundational idea in our autonomous tagging approach.

Table 3.2: Detection Performance with Tag Width Variation

% of Width Variation	C-Net		D-Net		Transcription	
	mAP	F1	mAP	F1	CRA	WRA
Decrease by 10%	85.06	91.20	88.69	92.41	91.90	80.94
Precise (0%)	91.41	95.08	92.77	95.38	93.61	86.80
Increase by 10%	91.13	95.02	92.32	94.59	93.24	86.14
Increase by 20%	90.04	94.35	92.06	94.38	93.07	85.90
Increase by 30%	89.39	93.85	90.73	93.73	92.71	84.72
Increase by 40%	87.44	90.08	89.04	90.42	92.39	82.28

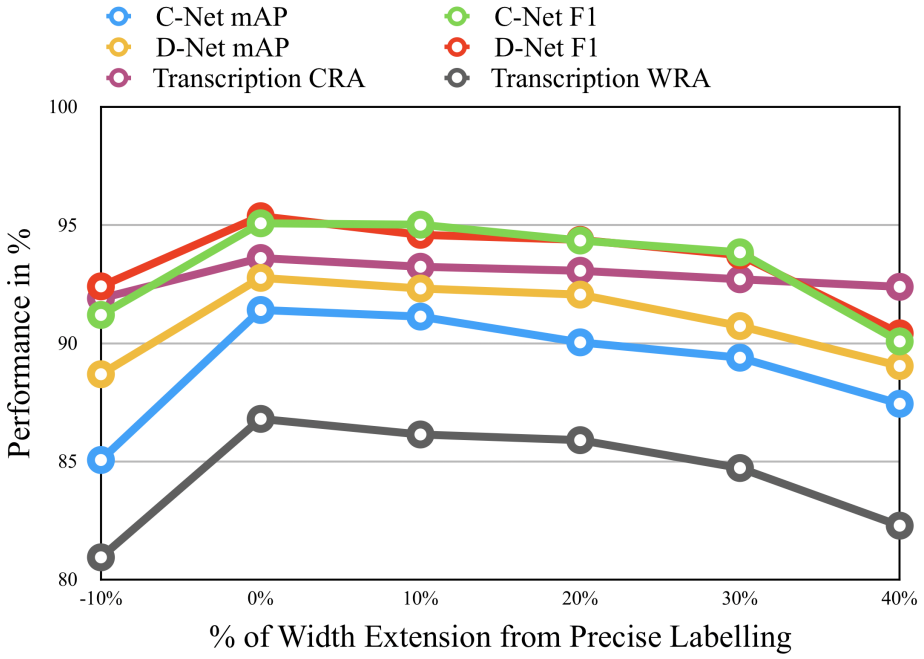


Figure 3.24: Plots of detection performance with tag width variation.

3.6.3 Basic Idea of Autonomous Tagging Process

The autonomous tagging process first estimates the locations of the characters inside a word and then extends the boundaries to an amount so that even allowing

for variabilities in handwriting, the target character is most likely to be somewhere inside that extended bounding box. This is possible since we already observed that extending bounding boxes of ground truth tags does not considerably impact the detection performance. The details of the process for initial location estimation and boundary extension for Bangla and Korean are presented in the following subsections.

3.6.4 Implementation for Bangla

The initial location estimation of the characters and diacritics for handwritten Bangla words are approximated from a machine printed version of the ground truth text. The process is explained with an example in Fig 3.25 for a three character word. Here W indicates the total width of the word composed by the three characters (including diacritics) with widths of X, Y and Z. The subscripts H, P and E represent the handwritten, printed and estimated character widths respectively. The width of the printed characters (including diacritics) are measured from the machine generated font, from which they are proportionately imposed on the handwritten word with a width extension factor of η. For this experiment η was chosen to be 20%, 30% and 40% of the initial estimated width reflecting the observation from our tag variance experiment described in Section 3.6.2. This created 3 images for each estimate all of which were used for training, which is analogous to the 3 augmented images created during training for the original manual-tag training [85]. Thereby, the number of training samples for both experiments are the same. All estimated widths are extended by $\eta/2\%$ on both sides except for the boundary characters, which are extended by $\eta/2\%$ only on the interior edge.

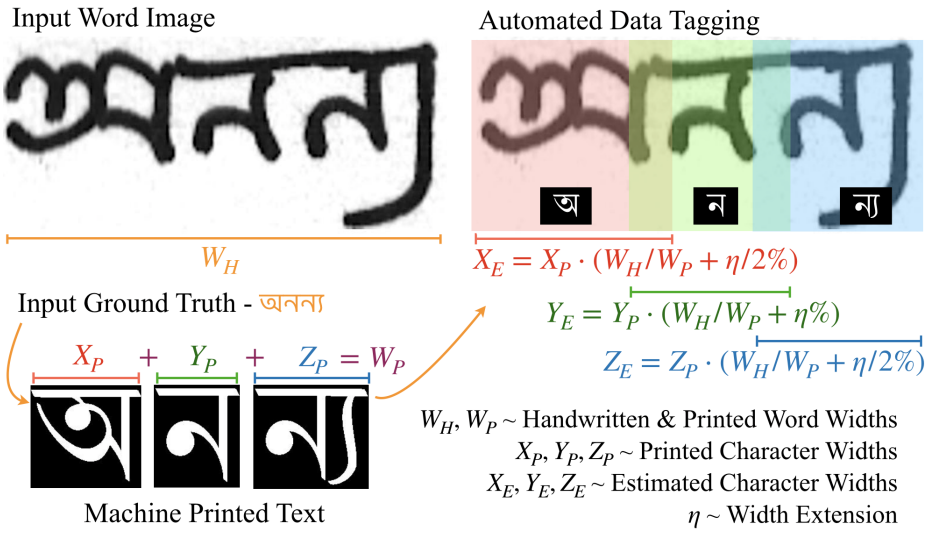


Figure 3.25: Example of autonomous tagging from a printed font for a three character Bangla word. Based on the character widths obtained from the machine printed text ($^*P$), the widths of the characters and associated diacritics in the handwriting are estimated ($^*E$).

The tagging and recognition work process is shown in Fig 3.26. This example shows the boundaries around the Bangla character 'ক' after autonomous tagging for the training. The object detection network is trained to locate the same character from a given boundary, but the boundary actually contains most of the target character and some extra parts around it. These parts can be a diacritic, or can be a chunk from the prior or next character or diacritic. Furthermore, these extra undesired elements can be anywhere (left, right, top or bottom) and will not have a fixed pattern. This phenomenon over the iteration of neural training prepares the network to treat anything different from the target character as arbitrary and not important for the decision. However there is a cost of training an object detection network in this format. The location of the character can not be precisely identified since it was not trained with accurate locations. Therefore, the network

can confidently predict the class, but only vaguely predict an area within which the character is located as shown in Fig 3.27. This is not a problem for handwriting recognition, since we only need the relative positions of characters or diacritics with respect to the others. This particular drawback doesn't impact the transcription accuracy by a noticeable amount.

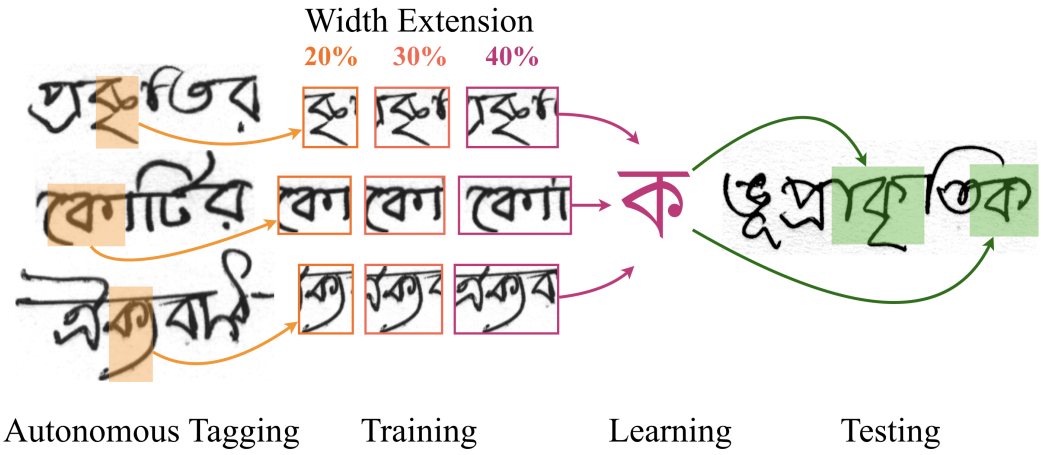


Figure 3.26: The schematic illustration of how autonomous tagging works. Each character is boxed with variable widths for training. The position of the learned character is shown in the test words.

For the printed font we used 'Akaash' which we empirically found has a nice match with typical handwritten shape proportions. Additionally most of the time people include a larger space before punctuation in handwriting than appears in machine print, therefore, we inserted one blank space before each punctuation mark to obtain a better estimate.

The difference in performance from autonomous tagging versus precise labelling actually comes from the eccentric property of human handwriting. No matter how good the initial estimate is or how robust the network performs with width tampering, there will be some cases where the autonomous tagging misses

the target character/diacritic completely or a major structural part of it. A similar situation can also happen from manual labeling since that process is very error-prone. In both cases, this issue can be solved by increasing the volume of the dataset. With more data, the ratio of proper labeling to mislabels gets higher and the network gets enough good samples to be effectively trained. Preparing larger quantities of data is significantly more convenient with an autonomous tagging framework than using manual annotation.

Figure 3.27: Detection from the networks trained with manual (left) vs. autonomous tagging (right).

3.6.5 Implementation for Korean

For Korean we used a much simpler approach for the initial character size and position estimation. This is related to the well-defined structure of the Hangul syllables. Unlike Bangla, the Korean characters can also appear above or below each other. We obtained a list from the Unicode foundation of how every Unicode Hangul syllable was composed of its three constituent parts: lead consonant, vowel, post-consonant. We identified whether the vowel was a vertical symbol that would appear on the upper right of the syllable, a horizontal symbol that would appear below the lead consonant, or a compound vowel that would appear with one component in both geometric places. Similarly it was determined if

there was no post-consonant, if the vowel was followed by a single Jamo or by a compound of two Jamos. The height (or width) was then divided into 1, 2 or 3 zones and that distance apportioned to those symbols with some additional buffer size. This resulted in the 8 different geometric structures of Hangul composites shown in Fig 3.28.

If there are two Jamos over the vertical (horizontal) span of the syllable, the initial zone height (width) estimate for each Jamo is 50% of the whole syllable height (width), like the structures shown in Fig 3.28 (a) through (e). Just like the Bangla character widths were increased, this base estimate of 50% is used, even if the specific Jamos do not have the same height (width). The estimate of 50% is increased for each zone to guarantee the handwritten Jamo is included, and to produce overlap between the Jamo zones. For cases where there are three Jamos, like the structures shown in Fig 3.28 (f), (g) and (h), the initial height estimate is 33%, which is then extended.

To expand our training set similar to with Bangla, we allowed three different extensions. For the two Jamo partition, the initial size estimate of 50% was increased to 60%, 70% and 75%. For the three Jamo partition, the initial size estimate of 33% was increased to 40%, 45% and 50%.

Machine printed fonts could have also been used for Korean the same way we did for Bangla to account for when the Jamos don't have equal heights or widths, but this approach is simpler and works well.

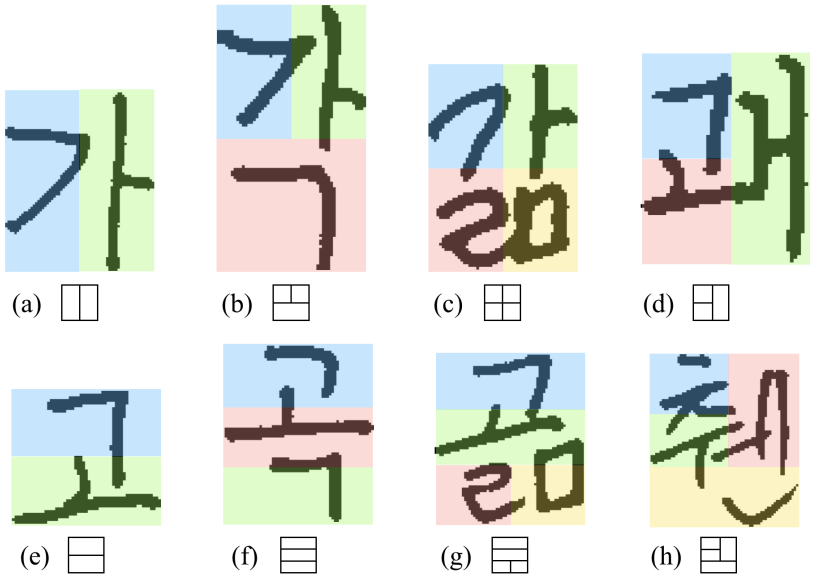


Figure 3.28: Initial estimated bounding boxes of the Jamos from a compound Korean syllable. Widths and heights are divided into 2 or 3 zones based on to which geometric structure from (a) to (h) it belongs.

We tested all the designs and tools we introduced in this chapter through a number of experiments. The setup and outcome of these experiments as well as the analysis comparing on results with other equivalent works (if available) are presented in Chapter 4.

CHAPTER 4

RESULTS AND ANALYSIS

In Chapter 3 we introduced our offline handwriting recognition framework as well as the supporting tools and algorithms like the Boise State dataset, autonomous tagging, etc. Also we explained the experimental arrangements to verify and validate everything we designed. In this chapter, we discuss the execution and outcome of these experiments as well as the implication and analysis from these results. The following is a list of the experiments that will be discussed in the rest of this chapter -

1. Benchmarking the isolated character set in the Boise State Bangla Handwriting dataset with other publicly available datasets using the isolated character recognizer described in Section 3.4.7.

2. Testing the proposed character spotting offline recognition system introduced in Section 3.2 for Bangla using the Boise State dataset as well as the three other Bangla datasets mentioned in Section 3.5.2. The details of this experiment were discussed in Sections 3.2.2 and 3.3.

3. Testing the proposed character spotting offline recognition system for Korean in the process described in Sections 3.2.3 and 3.3.

4. Comparing character spotting recognition performance when using autonomous tagging vs. precise tagging on the training data. Autonomous tagging was implemented both for Bangla and Korean as described in Sections 3.6.4 and 3.6.5 respectively.

5. Comparing character spotting and isolated character recognition performance between camera-acquired and scanned images.

4.1 Benchmarking Isolated Character Set in the Boise State Bangla Handwriting Dataset with Other Publicly Available Datasets

As we discussed in Section 3.4.2, a part of the Boise State Bangla Handwriting dataset is a collection of documents with isolated components which have handwritten samples of the basic characters, diacritics, numerals and some high frequency conjuncts. Upon launch of the Boise State dataset, we also prepared an isolated basic character recognizer to benchmark our dataset with three other similar public datasets [92, 93] as described in Section 3.4.7. Isolated character recognition is an important part of offline recognition research since alphanumeric characters in isolated form appear in many places such as postal headers, house addresses, flyers, notices, bank checks and tickets. Therefore, an isolated character recognizer facilitates automations of many tasks like sorting, filtering, etc. for these kinds of documents. This also presents the inter-compatibility among these datasets. In a conventional machine learning approach, we used features extracted with zonal pixel counts, structural strokes and grid points with U-SURF descriptors modeled with bag of features, details of which were explained in Section 3.4.7.

The other datasets we used with the Boise State dataset were CMATERdb 3.1.2, ISI handwritten basic Bangla characters and Banglalekha database (described in Section 2.4). A number of experiments were conducted using different combinations of all these datasets. First, we prepared three classifiers from different training data as follows:

1. Training set of CMATERdb 3.1.2 (12,000 samples, 240 per class),

2. Training and validation set of ISI handwritten basic Bangla characters (25,000 samples, 500 per class) and

3. A selected set of BanglaLekha database (60,000 samples, 1,200 per class).

These classifiers were then tested on the -

1. Test set of CMATERdb 3.1.2 (3,000 samples, 60 per class),

2. Test set of the ISI handwritten basic Bangla characters (12,858 samples, unevenly distributed) and

3. A randomly selected subset of the BanglaLekha database (5,000 samples, 100 per class).

After these experiments, we tested each of these three classifiers on the first 100 pages of the Boise State isolated component dataset (4,844 samples, unevenly distributed). Finally, a combined dataset was formed from the three training sets (97,000 samples, 1,940 per class) and tested on the isolated characters from the Boise State isolated Bangla character dataset. The outcomes of each of these experiments are presented in Table 4.1. The Boise State dataset was never used for both

training and testing simultaneously as it was not sufficiently large to partition at the first release.

Table 4.1: Isolated basic character recognition accuracy obtained from different training and testing sets

Dataset used for Training	**Dataset used for Testing**	**Recognition Accuracy**
CMATERdb 3.1.2 Training Set	CMATERdb 3.1.2 test set	92.87%
	Boise State character db	91.39%
ISI db Training and Validation Set	ISI db test set	93.10%
	Boise State character db	89.24%
BanglaLekha db (Selected Samples)	BanglaLekha db	96.80%
	Boise State character db	95.78%
Combined	Boise State character db	96.42%

As can be seen from Table 4.1, the recognition accuracies were consistently lower when the Boise State dataset was tested, but not by much. This is primarily because the similarity between the training and testing datasets is lost probably since our test set was cell-phone camera acquired, where the others were all scanned on a flat-bed scanner in 300/600 dpi. The classifier obtained from the combined training set of the these three datasets produced the maximum recognition accuracy of 96.42% when tested on the Boise State dataset. This not only supports the well known statement of "more data is better", it also shows that the Boise State dataset does not have any compatibility issues even when all these other datasets are simultaneously used.

We present in Table 4.2 some notable works on isolated Bangla character recognition along with the best recognition result we achieved. As can be seen, Roy et al. [101] reported 86.40% accuracy on the CMATERdb 3.1.2 dataset. Alif et al. [46] reported 95.99% accuracy on the same dataset, although the number of classes, training and test sets were not explicit. Our approach achieved 92.87% accuracy on this dataset. One of the best reported accuracies on the ISI dataset, reported by Bhattacharya et al. [83], is 95.84% using a two stage classification scheme. Our approach on this dataset produced an accuracy of 93.10% with a single stage classifier. On the BanglaLekha-Isolated dataset the highest reported accuracy was 95.10% by Alif et al. [46] using a convolutional neural network. This dataset is significantly larger than the others, therefore we didn't use the entire dataset to replicate their experimental setup, since our primary objective was to benchmark the Boise State dataset with their dataset. The obtained accuracy with a randomly selected portion of this dataset with our method was 96.80%, which is the maximum performance for Bangla isolated characters reported so far. Note that Alif et al. used 84 classes (including some high frequency conjuncts with the basic characters) and we used 50 classes of the basic characters only, therefore the performance scores are not directly comparable with each other in this case.

Table 4.2: Some notable research on isolated Bangla handwritten basic character recognition compared with the presented approach

Researchers	# Classes	Feature	Classification	Dataset Used	Test Set	Max Accuracy
Bag et al. [40]	50	Skeletal Convexity	LCS	ISI	500	60.60%
Bhowmick et al. [38]	50	Stroke Features	MLP	Private	4,500	84.33%
Roy et al. [101]	50	Directional Gradient features with ABC optimization	SVM	CMATER	3,000	86.40%
Rahman et al. [102]	49	Various Structural Attributes	TMS, BWS, FWS, MLP, MPC in a multistage	Private	N. A.	88.38%
Bhowmik et al. [103]	45	Wavelet Decomposition	Two stage HLA with SVM	Private	5000	89.22%
Bhattacharya et al. [104]	50	Shape Feature Vectors modeled with HMM	MLP	ISI	9,481	90.42%
Bhattacharya et al. [83]	50	Gradient Directions, Regional Pixel Counts	MQDF and MLP in two stage	ISI	12,858	95.84%
Alif et al. [46]	84	N. A.	Convolutional Neural Network ResNet-18	BanglaLekha	33,221	95.10%*
Presented Work	50	Zonal Pixel Counts, Stroke Patterns and SURF	SVM with Polynomial Kernel	CMATER, ISI, BanglaLekha,	4,844	96.80%

* Reported 95.99% on CMATERdb, # classes, training and test set information weren't explicit

The maximum classification accuracy of 96.8% that we achieved is the highest reported accuracy for isolated Bangla basic character recognition, not only among these datasets but also among all those reported in the literature. Although, we obtained the best result for this, the fundamental point was to see the compatibility of this character dataset with the others. Even though the acquisition processes are different and two of these three datasets were actually prepared with writers from a different country, this tiny experiment shows there were no notable issues with our dataset. This indicates for almost any experiment or approach all these datasets can seamlessly be used as resources without any kind of special treatment.

4.2 Offline Recognition Performance for Bangla with the Proposed Character Spotting Framework

The offline character spotting recognition framework we presented in Sections 3.2 and 3.3 has been the center of our research during all of our experiments. This was primarily designed to work for Bangla and the implementation process is explained in Section 3.2.2. The character spotting method we introduced is a simple and very effective way of achieving offline recognition. This method depends on segmented character level ground truth metadata, which is why we trained our recognizer networks (C-Net and D-Net) using only the Boise State dataset since the other public Bangla datasets do not have this. However, this is a segmentation-free recognition process and thereby can be tested with any other datasets. In fact, this is one of the best ways to assess the practical strength of a recognition system and how it might perform in real-life applications. Therefore, we tested our framework with the Boise State dataset as well as three other Bangla datasets as described in Section 3.5.2 and the outcomes are presented in this chapter. We also conducted a smaller experiment on the Korean script in order to demonstrate how this framework can be adapted other scripts as well. The results with Korean are presented together with our autonomous tagging system performance in Section 4.4.2.

4.2.1 Character Spotting Recognition with the Boise State Dataset

While using our character spotting recognition framework with the Boise State dataset, all the word images were resized to 600 pixels at its smallest dimension (usually heights). There were two batches of experiments used to evaluate the recognition performance – one using the first 150 camera-acquired essay pages [85] and the other using all 253 essay pages (both camera-acquired and scanned versions) and 70 conjunct word documents for training and testing.

Experiment 1: Using the first 150 Camera-Acquired Essay Scripts

In the first experiment, there were a total of 15,656 word images, 90% of which were used for training and the rest for testing. The 12,525 training images were quadrupled to 50,100 images with data augmentation (described in Section 3.3.3) and these were used for training both the C-Net and D-Net. The class distribution for the networks in this experiment is shown in Table 4.3. This table is the same as Table 3.4 but without the conjuncts written in blue since those were only present in the conjunct word documents of the Boise State dataset (Section 3.4.2).

Table 4.3: List of C-Net and D-Net Class Distribution when only the Essay Scripts from the Boise State dataset were used

C-Net Classes			**D-Net Classes**		
Basic	Conjunct	Punctuation	Diacritic	Basic	Conjunct
অ, আ, ই, ঈ, উ, ঊ, এ, ঐ, ও, ঔ, ক, খ, গ, ঘ, ঙ, চ, ছ, জ, ঝ, ট, ঠ, ড, ঢ, ণ, ত, থ, দ, ধ, ন, প, ফ, ব, ভ, ম, য, র, ল, শ, ষ, স, হ, ড়, ঢ়, য়, ং, ঃ, ৎ	ঙ, চ্চ, ঙ্গ, ঞ্ছ, ঞ্ঝ, ষ্ট, ষ্ঠ, ষ্প, ষ্ফ, ন্ত, ন্ন, দ্ব, ক্স, ক্ষ, ত্ত, ট্ট, ষ্ণ	‘, (কমা)’, ‘। (দাড়ি)’, ‘- (হাইফেন)’	া, ি, ী, ু, ূ, ে, ৈ, ো, ৌ, ্	ঁ	ঁ, ্র, ্য, ্ব

Table 4.4: Recognition Performance Scores obtained from Experiment 1 and 2

Network	**Performance Parameters**	**Scores from Exp 1 (150 Essay Scripts)**	**Scores from Exp 2 (All Essay & Conjunct)**
C-Net	mAP	0.8713	0.8815
	F1 Score	0.8961	0.9258
D-Net	mAP	0.9034	0.9034
	F1 Score	0.9317	0.9317
Word Recognizer	Precision	0.8825	0.8825
	Recall	0.8942	0.9124
	mAP	0.8842	0.9032
	F1 Score	0.8996	0.9265
	WRA	0.7564	0.8774
	CRA	0.8880	0.9480
	WRA (after spell check)	0.7848	N. A.
	CRA (after spell check)	0.9109	N. A.
	WRA (5-fold Cross Validation)	N. A.	0.8618
	CRA (5-fold Cross Validation)	N. A.	0.9437

The recognition performance of this framework is evaluated with individual mAP (from *Average Precision* or AP from individual classes) and $F1$ score of C-Net and D-Net as well as *Precision*, *Recall*, mAP, $F1$ score, WRA and CRA of the transcription (these parameters were defined in Section 3.3.6). The WRA and CRA are evaluated both with and without the spell checker (described in Section 3.3.5). All these performance scores are presented in Table 4.4 (column 3).

Histograms of the $F1$ scores for C-Net and D-Net are shown in Fig 4.1. The characters with a low number of occurrences in the dataset ended up having poor detection results, because the training simply wasn't enough to perform well. This is especially true if that character has a visual similarity to other classes. The lowest three $F1$ scores we obtained are for 'ঊ' (dirghô ū), 'এ' (e) and 'শ্চ' (shô + chô) of 0.66, 0.71 and 0.73 respectively. These are all from C-Net classes; the D-Net scores are relatively stable and more uniform with the worst $F1$ score at 0.81. We used 0.66 and 0.81 as the confidence thresholds for C-Net and D-Net detection as explained in Section 3.3.4.

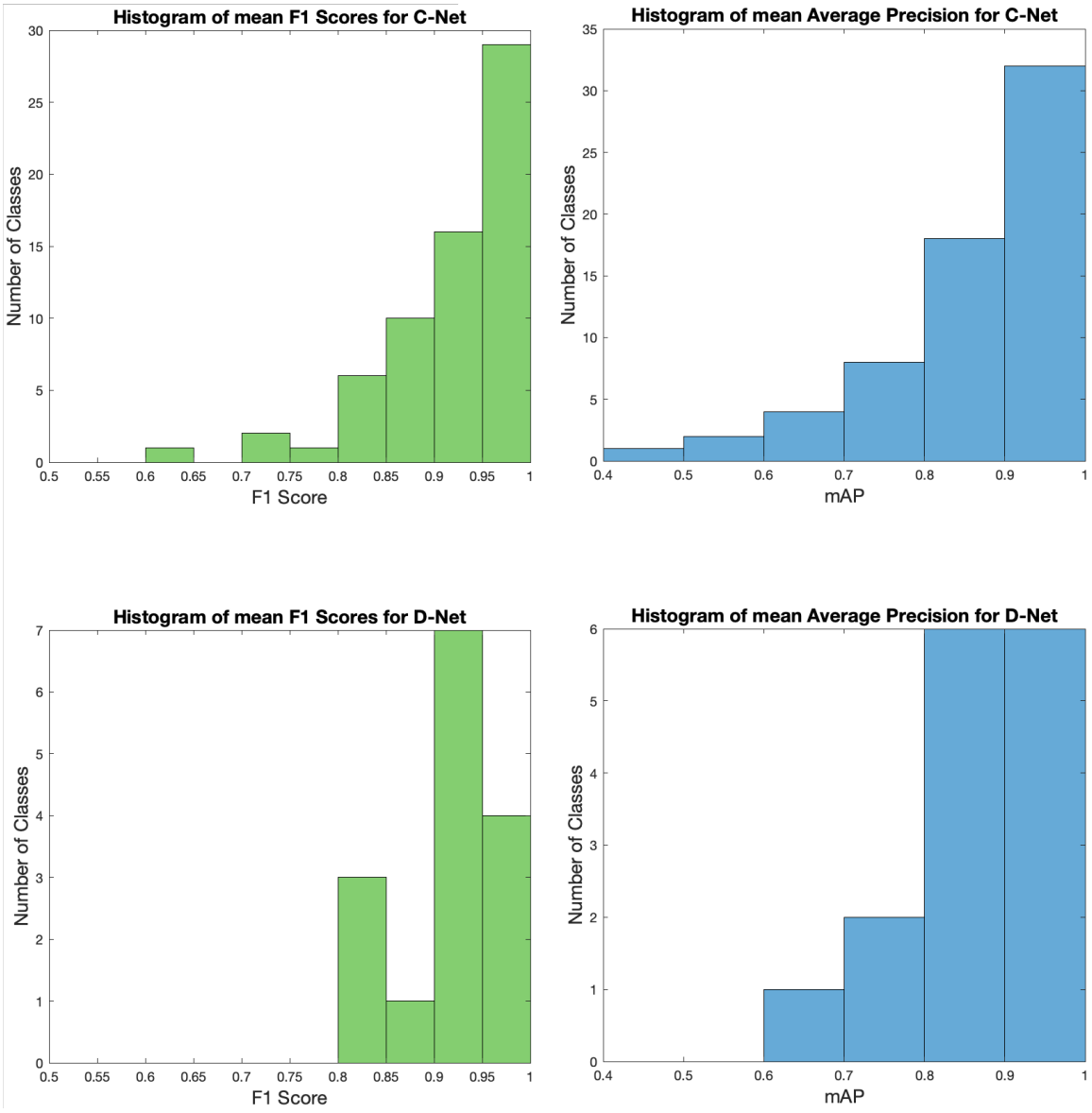


Figure 4.1: Histograms of mean Average Precision (mAP) and mean F1 scores from C-Net and D-Net detection results.

In order to measure the location accuracy of the character/diacritic spotting, we also measured the Intersection over Union or IoU as defined in Equation 3.3 with this experiment. The IoU is a standard evaluation metric used to measure

the accuracy of an object detector on a particular dataset. We found the average IoU for C-Net and D-Net to be 0.53 and 0.47 respectively. Values of IoU around 0.5 are usually considered poor, but in our approach this is quite expected since we are forcing the networks to be trained with additional diacritic (for C-Net) and character (for C-Net) components around as explained in Section 3.2.2. In fact, the C-Net/D-Net detected characters/diacritics were actually more precise than the location tags we used for training when compared to the actual location as shown in Fig 4.2. The blue box in this figure contains a character and a diacritic. This location was used for both C-Net and D-Net training. The green box shows the precise location for the diacritic in this query and the orange box shows the D-Net detected location for that diacritic. As can be seen, the D-Net detection (orange box) is actually a better estimation of the precise diacritic location (green box) than the location used used for training (blue box). Therefore, with this sequential character spotting strategy the IoU scores are expected to be low and that does not impact the recognition performance since the compilation entirely depends on the relative locations of characters or diacritics, not on their accurate locations.

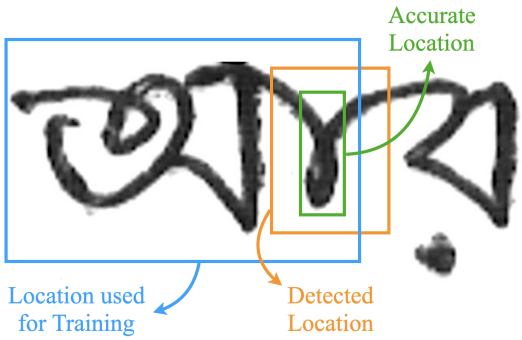


Figure 4.2: Case demonstration of why IoU with sequential character spotting approach is low. Green shows the exact location of the target diacritic. Blue shows the location used for sequential C-Net/D-Net training which includes the associated consonant with the diacritic. The orange box shows the D-Net detection which is closer to the accurate location than training location.

Experiment 2: Using all Essay and Conjunct Word Documents

The next experiment was done using all 253 essay scripts (both camera-acquired and scanned versions) and the 70 conjunct word documents of the Boise State dataset. This translates to a total of 60,157 words from which a 90% and 10% split of the data taken from an evenly distributed from the camera-acquired and scanned essay scripts as well as the conjunct word documents was made for training and testing respectively. We also used a 80% - 20% training and testing combination (similarly evenly distributed) to obtain a 5-fold cross validation result on WRA and CRA. The experiment process was the same as the experiment 1 except that we didn't use the data augmentation and spell checking with this setup. Also, the D-Net classes remained the same, all the new conjuncts from the conjunct word documents were added to the C-Net as shown in blue text in Table 3.4. The results are presented in Table 4.4 (column 4).

As expected, the score went higher by using more data even after introducing a lot more classes than the first experiment. Furthermore, this result represents the true offline handwriting performance for Bangla since it covers almost every element and variability that comes with this script. One of the major problems with the overall detection was with the high number of false positives with the diacritic 'া-কার' (AA-kar). While in machine print this has a distinct shape, for handwriting, in most of the cases, this just becomes a vertical line, hence, it is easy to false detect this inside any other characters/diacritics which include such a straight line. Therefore, one major room for improvement is to have a special treatment for this diacritic.

4.2.2 Character Spotting Recognition on other Bangla Datasets

To test our framework with other Bangla datasets, we trained the C-Net and D-Net with all 60,157 words from the essay scripts and conjunct word documents in the Boise State dataset, without any data augmentation. These networks are then tested on the CMATERdb 1.1.1, Indic Word Dataset and REID2019 as introduced in Sections 3.5.2, 3.5.2 and 3.5.2. We tested our character spotting recognition framework on these datasets as follows:

- For CMATERdb 1.1.1 we manually transcribed the first 25 pages from this dataset (since the transcription was not available) and used our recognition framework with the word coordinates of those pages provided by the CMATER group.

- For the Indic Word Dataset we only used the test set which contains 3,856

word images. Word transcription metadata was available with this dataset.

- For the REID2019 dataset, the first 11 pages from the evaluation set were used for testing. Word transcription metadata was available with this dataset too.

We measured the CRA and WRA (defined in Section 3.3.6 from the transcription result. No spell checker was used in any of these experiments. The results are presented in Table 4.5.

Table 4.5: Recognition Performance with other Bangla datasets

Dataset Used	**CRA**	**WRA**	**WRA (Top Reported)**
CMATERdb 1.1.1 [79, 78]	92.36%	82.27%	N. A.
Indic Word Dataset [97]	89.97%	78.21%	88.19% [97]
REID2019 [98]	93.08%	83.62%	$<$ 80% [98]
Boise State Dataset	94.80%	87.74%	87.74%

As seen, although the scores are lower than when tested with the Boise State dataset (Table 4.4), it does not deviate much and the total in each case is still well above the standard of reliability. There is no transcription level work reported using the CMATERdb 1.1.1 dataset. The best reported work for the Indic Word Dataset is presented by Mukherjee et al. [97]. They obtained a WRA of 88.19% using a fused LSTM network using a whole word recognition method (explained in Section 2.3 and thereby restricted recognition to only the words available in their dataset. In contrast, our WRA of 78.21% on this dataset is using a recognition process that is not limited to a fixed set of words.

Surprisingly, the best performance we obtained out of these three datasets is with the REID2019 dataset, which is not even handwritten, rather machine printed historical documents. This dataset was used for a competition on recognition of early Indian printed documents in the International Conference on Document Analysis and Recognition (ICDAR) in 2017 and 2019. All the OCR results submitted to this competition were below 80%, while we obtained a CRA of 93.08% with this dataset in spite of the fact that our recognition framework has never seen any machine printed text during training. Overall, the outcome of this experiment strongly suggests that our presented framework is robust enough to be used for unconstrained handwriting recognition in real life applications.

4.3 Performance Comparison between Camera-Acquired and Scanned Images

Since the Boise State dataset has both scanned and cell-phone camera acquired versions of the essay scripts and isolated element pages, we experimented to see how the recognition performance varies based on just the acquisition source difference [105]. A visual difference between these two digitization sources is shown in Fig 4.3. A flat-bed scanner is considered to be the ideal source of document data acquisition. It usually offers better quality images with uniform lighting and higher resolution. Camera-based acquisition suffers from noise, blur, perspective distortion, jpeg compression and many other complex artifacts that arise from the lighting condition as well as interaction of the background and foreground. As a result, scanned images are the primary choice for most handwriting document

datasets. For example, CMATERdb [78, 79], ISI db [83] and Banglalekha-Isolated [84] are the most widely used datasets for Bangla offline handwriting recognition, and they all contain only scanned images. Not only for Bangla, most other popular offline handwriting datasets such as MNIST (English) [106], iAM (English) [107], IFN/ENIT (Arabic) [108] and ETL Kanji (Chinese) [109, 110] etc. are also formed with scanned documents only.

Although a scanner produces better quality document images, a camera-based system offers better convenience, which is vital for many applications using offline handwriting recognition. Digital cameras are almost always embedded in cellular phones and they are mobile as well as easy to use. In recent years they not only have seen a significant increase in performance, they have also become cheap and accessible all over the world. In some applications, such as with sensitive and fragile historical documents, it is often preferred to do a contactless acquisition. Because of these reasons, researchers as well as industries are now shifting towards camera-based document analysis. But there is no specific work that compares the recognition performance for handwriting documents just based on these two acquisition sources. This is partially due to the lack of available dataset.

Here, our goal is to see how the image acquisition method affects the ability of a classifier to recognize the text. To see the effects, two different types of base frameworks were used: one offline Bangla handwriting recognizer with character spotting [85] with the process described in Section 4.2.1 and one Bangla handwritten digit recognizer with an SVM classifier[93] using the process described in Section 4.1.

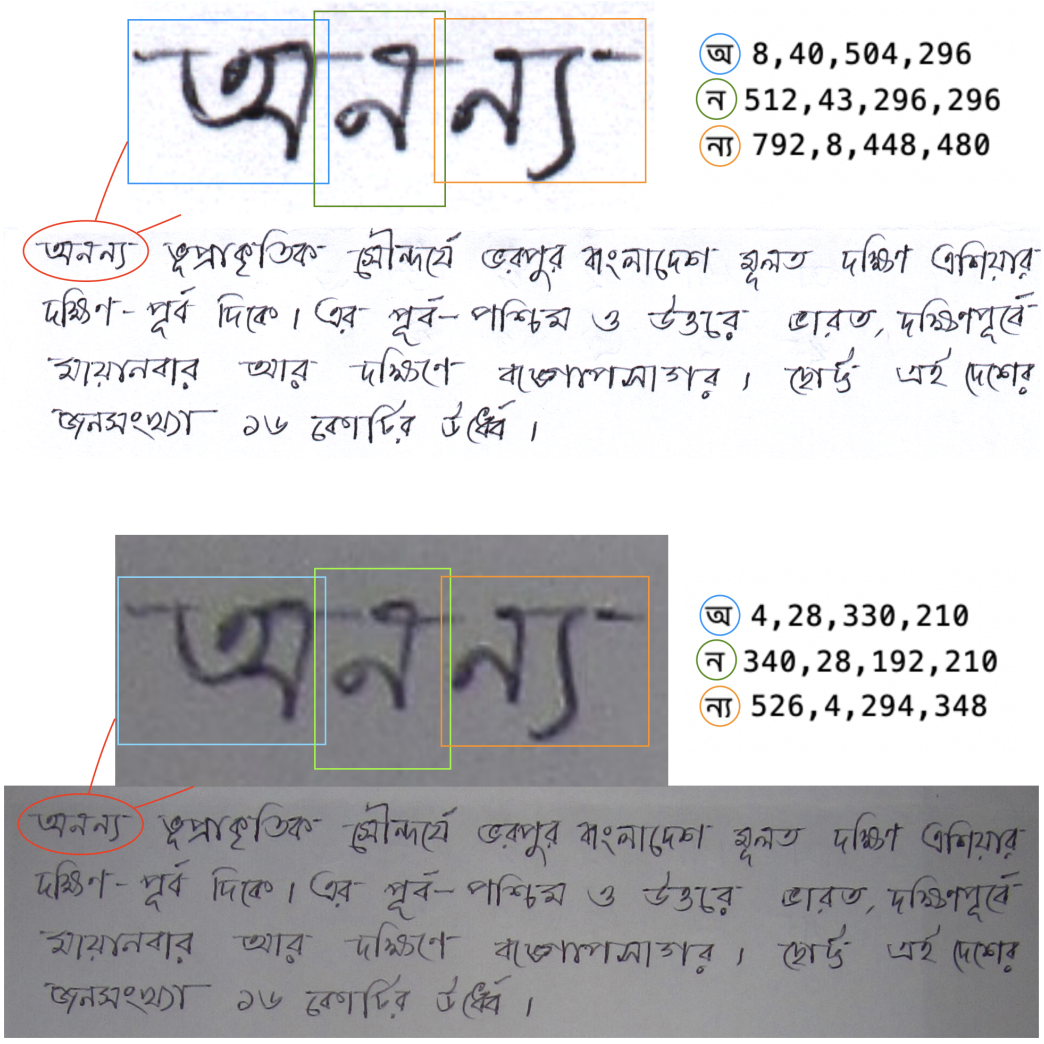


Figure 4.3: Sample pair of scanned and camera-acquired document image with ground truth bounding box tagging information from the Boise State Bangla Handwriting dataset [111].

The first framework we used is our unconstrained offline character spotting handwriting recognizer presented in Section 3.2.2. We used exactly the same sequence of data as the first experiment of Section 4.2.1. For the second framework, we used the isolated character recognizer as presented in Section 3.4.7. For this we used only the handwritten digits instead of the basic characters. This is a 10 class problem where each class contains approximately 250 instances. The outcomes are recorded with a 10-fold cross-validation using the same digits in each fold, whether it be scanned or camera acquired for all the experiments. There are in total 4 individual experiments for each network of both frameworks, training and testing on Camera-Camera, Scan-Scan, Camera-Scan and Scan-Camera. The character/diacritic detection performance is evaluated using mAP, $F1$ scores and CRA (defined in Section 3.3.6) as shown in Table 4.6. The digit recognition performance is measured in terms of recognition accuracy, shown in Table 4.7.

Table 4.6: Detection Results from Different Acquisition Sources for the Character/-Diacritic Spotting Networks

Training Set	**Testing Set**	**C-Net**		**D-Net**		**CRA**
		mAP	**F1**	**mAP**	**F1**	**(1 - CER)**
Camera Images	Camera Images	83.36%	84.22%	85.30%	86.81%	84.75%
Camera Images	Scanned Images	80.08%	79.93%	83.26%	84.02%	82.00%
Scanned Images	Camera Images	86.95%	84.56%	88.94%	91.11%	85.54%
Scanned Images	Scanned Images	88.77%	87.61%	90.28%	93.66%	86.62%

Table 4.7: Recognition Results from Different Acquisition Sources for Handwritten Digits

Training Set	**Testing Set**	**Accuracy**
Camera Images	Camera Images	94.51%
Camera Images	Scanned Images	90.64%
Scanned Images	Camera Images	94.72%
Scanned Images	Scanned Images	95.73%

As seen in Table 4.6 for the detection networks, the performance is better when scanned images are used for both training and testing than when the camera acquired images are used for both training and testing, but only by approximately 5%. Since the scanned images weren't manually tagged with ground truth information, rather transferred using a geometric transformation, some of these images are slightly clipped or over extended from what was expected. A couple of sample scenarios are shown in Fig 4.4. Therefore, the results could have been a bit better with the scanned images if all were tagged manually.

It is not surprising that the highest recognition accuracy occurs when the networks are trained on scanner acquired images and also tested on scanner acquired images. Likewise, it is not surprising that the lowest performance is when the network is trained on the lower quality camera acquired images and tested on the higher quality scanner acquired images. The surprise is that the overall accuracy for the training on scanner acquired images and testing on camera acquired images surpasses training and testing both on camera acquired images. Also, it can be seen that the detection results decrease slightly (roughly 3%) when the acquisition source is different from training to testing. For the C-Net, the $F1$ score is about

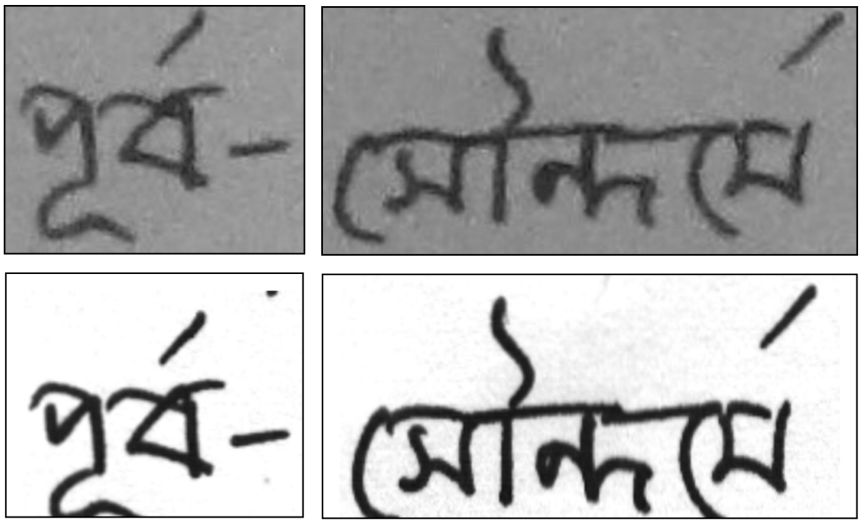


Figure 4.4: Examples of the clipping that occurred during ground truth tag transfer from camera-acquired (top) to scanned (bottom) images using a geometric transformation

the same regardless of training source when tested on camera acquired images. A couple of samples, where training on camera images failed but training on scanned images was successful on camera test images are shown in Fig 4.5.

The same trend follows with the handwritten digit recognition framework as well, but by even narrower margins as shown in Table 4.7. The scanned image based training and testing outperforms the camera image based one by just over 1%. Training with camera images has the lowest performance when tested on scanned images. Also, training with scanned images still surpasses training with camera images when both were tested on camera images, but in this case by only 0.2%. The stroke pattern features, which were used as one of the feature extractors for the digit recognition framework, diminishes the differences between a scanned and camera image, which may be one of the reasons that here the change in acquisition sources is not making as big of differences as with the other framework.

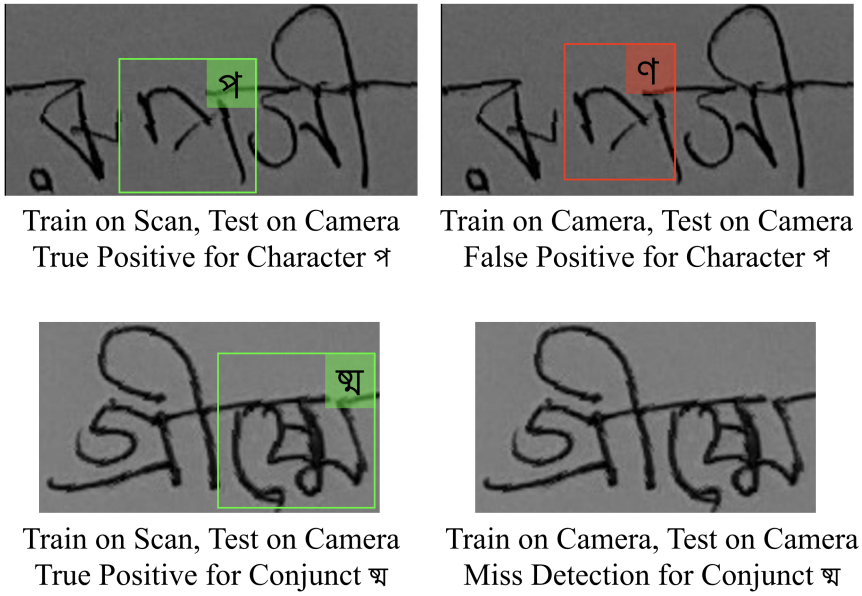


Figure 4.5: Sample cases of camera-acquired test images where training on camera-acquired images resulted in false (top-right) and miss (bottom-right) detection, but training on scanned images was successful (top left and bottom left) in both cases.

One reason for the trend in outcomes of the scanner-camera outperforming the camera-camera as train-test combinations, is most likely the fact that the camera-acquired images from the Boise State Bangla Handwriting dataset were acquired using multiple cell-phones in various lighting conditions. As a result, they do not have any advantage of internal similarity and vary quite a lot in terms of resolution, sharpness, illumination, lens quality and shooting angle. While this is not optimal, this is in fact the likely scenario for many applications of offline handwriting recognition with camera acquired images. Thereby, it turned out that the scanned image-based networks are simply better trained than the camera-based ones with the same diversity of handwriting. This is one of the reasons that we digitized the conjunct word documents of the Boise State dataset using only a flat-bed scanner.

From this we conclude it is advantageous, although not by a big margin, to train a network with higher quality images even if the application area is targeted toward images with lower quality. Although it might not be similar for every other experiment with different frameworks and/or datasets, the pattern we found is likely to appear in other cases.

4.4 Autonomous Tagging Performance on Bangla and Korean

The autonomous tagging concept as introduced in Section 3.6 is an automated process we developed to draw bounding boxes at the character level on handwritten word images using just the word level ground truth information. This method can be used with our character spotting based offline recognition approach. The fundamental objective is to save a great deal of time and manual labor that is needed to prepare a dataset with character level ground truth annotation and thereby accelerate the process of development. This technique can be used for almost any alphabetic writing system. Here, we demonstrated the process with Bangla and Korean, described in Sections 3.6.4 and 3.6.5. The recognition performance obtained with autonomously tagged datasets compared with what we obtained with manual tagging for these two scripts as well as the significance of this experiment are presented in the following subsections.

4.4.1 Autonomous Tagging for Bangla

The detection performance with the autonomous tagged dataset is measured with mAP, $F1$ score, CRA and WRA as in Section 3.3.6. These scores are then

compared with the performance obtained from precisely hand-labeled data. Table 4.8 shows all the results. As seen, the overall detection performance with autonomous tagging obtained from C-Net and D-Net decreased by roughly 2% from precise tagging. We consider this performance loss to be a worthwhile compromise considering the time and effort saved during the process. Furthermore, the results are on training sets of the same size. It is much easier to use more training data with autonomous tagging which will likely increase the performance. Also reported in Table 4.8 is the manual tagging performance (without the spell checker) reported in [85] and described in Section 4.2.1 as Experiment 1. Despite following the exact same process, the previous performance scores are lower than the current experiment. This is because, after that publication we discovered and fixed some data tagging errors. This illustrates that the recognition rates can be affected more by the human errors that occur while attempting precise manual tagging, than by the imperfection from the autonomous process. In fact, the overall detection performance was improved by roughly 2% with this experiment using auto-tag over our first attempt using manual labeling with inadvertent errors.

Table 4.8: Recognition Performance with Autonomous and Manual Tagging for Bangla

Method of	**C-Net**		**D-Net**		**Transcription**	
Data Tagging	**mAP**	**F1**	**mAP**	**F1**	**CRA**	**WRA**
Manual (Reported in [85])	87.12	89.61	90.27	93.18	88.82	75.55
Manual (Current)	91.41	95.08	92.77	95.38	93.61	86.80
Autonomous Tag	89.96	92.35	91.25	93.76	91.12	80.06

4.4.2 Autonomous Tagging for Korean

The Korean recognition performance is measured as Jamo Recognition Accuracy or JRA (equivalent to CRA) and compound Syllable Recognition Accuracy or SRA (equivalent to WRA). Our experiment uses a subset of the PE92 dataset both for training and testing. For training, we used 130 classes of syllables from the PE92 training set (each class contains 80 to 88 instances) chosen in a manner that every target Jamo shown in Fig 3.5 appears relatively evenly in our training. After being trained with autonomously tagged data from this training set, we tested the K-Net performance on 470 high frequency Hangul syllables (Appendix A) from the PE92 test set. The list of high frequency syllables was prepared from a syllable count in the 1000 most commonly used Korean words [112]. We also manually tagged a small portion of the training set (10 samples each from the 130 syllable classes). A trained network from these manually tagged data was used with the same test set to compare the performance.

The result from recognizing when K-Net was trained with auto and manually tagged characters is presented in Table 4.9, along with two other best scores achieved for this same dataset. As seen, using a large quantity of autonomously tagged data can outperform a smaller amount of precisely tagged trained data. While [52, 53] are able to achieve higher recognition rates, they devoted considerably more time to tune their systems for this dataset. They also limit their results to the 2350 syllable classes effectively implementing a spell checker, which our system does not.

The other results presented in Table 4.9 are not directly comparable to our

Table 4.9: Korean Recognition Results on PE92 Dataset

Researchers	**Methods**	**JRA**	**SRA**
Park et al. [52]	MQDF	N. A.	85.99%
Kim et al. [53]	DCNN	N. A.	92.92%
Presented Approach (Using a Subset of PE92)	Character Spotting (Autonomous Tagging)	91.22%	84.66%
	Character Spotting (Manual Tagging)	86.64%	79.23%

reported accuracy, since we used only a fraction of the PE92 dataset for training and testing. Still it can clearly be conjectured that our approach could achieve a high performing offline recognition rate for any script using a very small amount of data. Furthermore, unlike Bangla, we didn't use any post processing steps to assure that the detected combination of Jamos is permissible in the script. Even though vowel position was used in the tag generation process, it was not applied to the detection results. Such post processing steps would very likely improve the recognition performance.

4.4.3 Significance of the Autonomous Tagging Experiment

We demonstrated in Sections 4.4.1 and 4.4.2 how the autonomous tagging and the character spotting framework can work together for any script. Most Abugida scripts are structurally similar to each other and therefore, handwriting recognition for many Indo-Aryan scripts can be achieved just like Bangla. For Hangul, we could have done the autotagging using machine printed fonts for estimation, but we opted for a simpler approach because its well defined geometric structure allows us to do so. Even though we used a very small dataset for Hangul mainly to

demonstrate how our process can be applied to any script, the performance of the system was still great. Many other scripts with more complex structures don't even have labeled datasets. Therefore such a demonstration of how a working offline recognition process can be achieved with a very small dataset and bare minimum manual effort is a clear step forward in this field.

Modern technologies like Deep Convolutional Networks allow us to create end to end systems requiring minimum script specific processing. The only bottleneck that remains is to create and process a dataset. As presented here, the autonomous tagging takes away most of the difficulties involved in achieving this. In a simple view, this framework allows a dataset to be produced from just a collection of handwritten images with transcripts. Many such datasets for scripts like Latin, Arabic, Devanagari, Gurmukhi, Gujarati, etc. exist today while still no dependable or robust offline recognition is available for most of these. Although, there is a lot of room to improve this presented framework, we believe this simple idea has the potential to revolutionize the entire offline handwriting recognition field.

CHAPTER 5

CONCLUSION AND FUTURE DIRECTIONS

The problem of handwriting recognition has existed for more than half a century. Popular scripts like Latin have seen a lot of development, while many other scripts are mostly untouched. Here our attempt was to solve this problem altogether, for any script. Some of the important achievements our of this research are listed below.

1. **The design and development of an offline recognition system which is robust and flexible enough to be used with any alphabetic script.** We achieved a character recognition accuracy of almost 95% with Bangla using this approach. This is not only the first reported work for unconstrained Bangla handwriting recognition, but also an extremely well performing one when compared with the achievements of other scripts. The robustness of this system was tested with three other datasets with different structures. The flexibility was validated by transforming the framework to work for Korean. This is a thorough demonstration of a very powerful offline recognition method which can rapidly accelerate the growth of this field.

2. **The Boise State Bangla Handwriting dataset.** This is a free and easy to

use dataset that comes many unique and useful features to support many kinds of offline recognition research.

3. **The process of autonomous tagging, which makes the preparation of offline datasets for the training of the character spotting approach significantly easier.** This idea also works for different scripts which was demonstrated for both Bangla and Korean scripts. We demonstrated how autonomous tagging is equivalent to or even better in some cases than manual tagging. The character spotting framework combined with the autonomous tagging process is an incredibly powerful tool that can revolutionize the entire offline handwriting recognition field.

4. **The demonstration of how training with higher quality images is better even if the application field uses low quality data.** This is found from when we experimented with scanner and camera-acquired data to compare recognition performance. Although this was a small scale experiment and the statement might not be true for every case, it still opens up the opportunity for further research and questions many existing approaches like degradation models whether should or should not be used.

There is room for improvement and expansion in many of these presented tools, frameworks and experiments. For example, as pointed out in Section 4.2.1, there are a high number of false positives with the diacritic 'া-কার' (AA-kar) in our character spotting approach with Bangla. This diacritic being just a vertical line when handwritten is highly prone to get a false detection inside any other character/diacritic with a straight line in their structure. A possible solution to

this problem can be detecting this diacritic using a separate algorithm such as our stroke feature developed for the isolated character recognition as described in Section 3.4.7. Or a separate classifier could be designed only to look for its presence. Any improvement in the Bangla'া-কার' (AA-kar) detection can significantly improve the D-Net detection performance and thereby the overall transcription result.

In our experiments with the external datasets for Bangla as explained in Section 4.2.2, we only used a fraction of these datasets for testing. These datasets could also be for training which will make the C-Net and D-Net familiarized with even more samples of Bangla handwriting styles and therefore make the recognition system even more robust. Since the character level ground truth location tagging is not available with these datasets, we will have to use the autonomous tagging from the word level ground truth information. The Indic Word Dataset already has word level ground truth information, the CMATERdb 1.1.1 dataset does not and thereby will have to be manually transcribed before being trained with autonomous tagging.

Right now, the character spotting approach works only on word images and therefore is dependent on a page to be segmented to words because of the way we used the Region Proposal Network (RPN). As discussed in Section 3.3.2, the number or anchors or proposals for regions where the object lies is set to 64 for all of our experiments. This means 64 regions inside a given image are proposed during training or assessed during testing. This number works nicely when we are looking for characters from a given word image (or Jamos inside a syllable), but does not provide enough anchor points to find characters in a document page. In

theory, it is possible to increase the number of region proposal from 64 to a much bigger value so that all possible regions for characters in a document page can be proposed during detection, but this would make the training process much slower and thereby it is impractical. One approach is to isolate words from a document page, which is much easier than isolating characters from words. A possibly better approach to be able to transcribe a document page as a whole would be to slide a window across the document page, detect characters at each window position during the sweep and assemble the results into words and text lines. This is expected to work perfectly with character spotting since the networks are well trained to ignore additional components around the target classes. An intermediate solution is to segment a bigger chunk than words, such as text lines or a group of words and collect the locations/classes of the character components, which will bypass the necessity of having word coordinates in a document under test. This remains as a scope of future experiment to further improve the convenience factor of the character spotting framework.

The Boise State dataset was developed from the contributions of volunteers primarily from the academic environment. In the future, it would be good to add more handwriting samples to this dataset to even out the demographic distribution from what we currently have as shown in Fig 3.16. As the dataset grows with larger demographics for each category, it will be possible to observe how recognition performance varies with different group of people.

Our experiments with Korean as presented in Section 4.4.2 were brief and were primarily designed to demonstrate the potential of this method to operate successfully on different kinds of scripts. In the future the Korean recognition framework

could be expanded by using more data for training and further tuning the autonomous tagging by using machine print fonts to estimate the partitions. Also we didn't use any grammar-based correction rules after the K-Net detection. Implementing few basic rules like discarding invalid Jamo combinations or sequences or assessing empty spaces which can be analyzed later with a spell checker can vastly improve the recognition performance as it did for us with the Bangla script (via the process described in Section 3.3.4). Another opportunity is to start the K-Net training using the C-Net or D-Net parameters instead of VGG-16. VGG-16 is primarily designed to recognize objects from the ImageNet dataset which contains 1000 different classes of images with real life objects such as people, birds, flowers, furniture, etc. Although we trained the C-Net and D-Net using the VGG-16 network weights with transfer learning, currently C-Net/D-Net have better structural similarities with K-Net than VGG-16 in terms of the number of classes, the pattern of target objects, etc. Therefore, this process can result in faster and better learning and therefore improve the whole Korean recognition performance.

The offline recognition framework we designed is flexible to not only scripts, but also flexible many of the underlying tools and technologies we used. For example, we used a Faster RCNN for object detection and pre-trained VGG16 network weights to begin with, but the system is not dependent on any of those. Therefore, there is room for further experimenting with other technologies that might improve the overall performance.

Most of our experiments didn't use any post-processing. Applying grammar based corrections or language modeling can improve the recognition performance even further. The situation is the same with pre-processing too. We noticed using

compressed images can significantly reduce the training time without sacrificing any substantial amount of performance, but we did not do enough experiments to draw proper conclusions on this topic and thereby it remains a topic of future research.

Another future goal is to transform and test this framework on more new scripts. Many scripts like Arabic, Latin, Devanagari, etc. already have handwriting datasets with ground truth information at least at the word level and thereby can be immediately applied with autonomous tagging and character spotting with very little effort. For other scripts with no such datasets, we would have to depend on volunteers to collect and transcribe handwriting samples before applying these processes. Even with these scripts, the overall development is a manageable amount of effort since our frameworks perform well with small amount of data and the autonomous tagging method only needs word level ground truth data.

In this era, when technologies like Deep Learning are solving almost every problem that is thrown at it, it is surprising to see how poor the status of offline handwriting recognition is. Here, we presented a system that works for Bangla and Korean, but the fundamental contribution to this field is the idea of character spotting we demonstrated. I believe, this approach brings the decades old problem of offline recognition very close to a solution.

Much of this work has been subjected to peer review and presented at top international conferences. The following is a list of them.

1. N. Majid and E. H. Barney Smith, “Introducing the Boise State Bangla handwriting dataset and an efficient offline recognizer of isolated Bangla characters”

in 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR). IEEE, 2018, pp. 380–385, [93].

2. N. Majid and E. H. Barney Smith, "Segmentation-Free Bangla Offline Handwriting Recognition using Sequential Detection of Characters and Diacritics with a Faster R-CNN" in International Conference on Document Analysis and Recognition (ICDAR), September 2019, [85].

3. N. Majid and E. H. Barney Smith, "Performance comparison of scanner and camera-acquired data for Bangla offline handwriting recognition," in 2019 Workshop on Camera-Based Document Analysis and Recognition (CBDAR), vol. 4. IEEE, 2019, pp. 31–36, [105].

4. N. Majid and E. H. Barney Smith, "Autonomous Data Tagging for Offline Handwriting Recognition: Tested with Bangla and Korean Scripts" submitted for 2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR).

APPENDIX A

LIST OF HIGH FREQUENCY KOREAN SYLLABLES OBTAINED FROM THE 1000 MOST COMMON KOREAN WORDS

'다' '하' '이' '지' '기' '나' '아' '가' '그' '리' '자' '시' '어' '사' '대' '오' '전' '정' '동' '제' '내'
'마' '장' '주' '로' '모' '보' '생' '인' '르' '구' '들' '라' '무' '바' '부' '우' '일' '치' '경' '관' '니'
'러' '분' '조' '간' '거' '계' '고' '과' '국' '만' '상' '서' '식' '의' '한' '달' '말' '문' '물' '미' '발'
'방' '비' '선' '세' '신' '여' '음' '작' '현' '회' '히' '각' '건' '공' '교' '늘' '래' '성' '수' '실' '안'
'언' '업' '요' '원' '위' '재' '중' '차' '통' '편' '표' '학' '해' '결' '금' '날' '남' '노' '데' '도' '두'
'명' '민' '배' '버' '설' '소' '스' '연' '예' '있' '저' '적' '직' '화' '개' '것' '년' '느' '능' '단' '돌'
'등' '때' '런' '려' '반' '번' '불' '빠' '산' '살' '새' '술' '심' '알' '야' '얼' '역' '올' '용' '임' '체'
'키' '태' '터' '행' '향' '강' '걸' '곳' '당' '드' '듯' '떠' '또' '력' '루' '른' '름' '맞' '머' '목' '못'
'변' '쁘' '쓰' '양' '없' '열' '영' '운' '울' '웃' '월' '유' '입' '종' '책' '타' '품' '피' '할' '형' '후'
'감' '갖' '같' '격' '견' '근' '길' '까' '끝' '너' '네' '녀' '놀' '농' '놓' '누' '눈' '님' '담' '더' '되'
'디' '따' '람' '랑' '렇' '레' '련' '론' '료' '류' '막' '많' '맛' '멀' '면' '뭐' '받' '벌' '법' '붙' '삼'
'센' '속' '손' '씨' '악' '약' '얘' '엄' '외' '움' '으' '은' '잘' '잠' '쟁' '점' '족' '존' '좋' '죽' '준'
'줄' '즐' '질' '집' '찾' '최' '추' '커' '트' '특' '티' '파' '팔' '펴' '프' '필' '험' '혼' '확' '환' '활'
'힘' '갈' '갑' '값' '걷' '겁' '게' '겨' '겪' '곧' '광' '괜' '군' '굴' '귀' '규' '극' '글' '깊' '깝' '깨'
'꺼' '께' '껴' '꼭' '꽃' '꾸' '꿈' '뀌' '끌' '끔' '끼' '낌' '난' '낮' '낳' '냥' '넓' '넘' '넣' '녁' '놈'
'높' '눕' '뉴' '는' '늦' '닦' '닭' '답' '던' '독' '돈' '둘' '뒤' '든' '듣' '딸' '땅' '떤' '떨' '떻' '뜻'
'랗' '램' '럼' '럽' '렵' '령' '례' '롭' '롯' '룹' '룽' '린' '림' '맙' '맡' '매' '먹' '먼' '며' '몇' '몸'
'묻' '믿' '및' '밑' '밖' '밝' '밤' '밥' '벽' '별' '병' '본' '봄' '브' '빛' '빨' '빼' '뿐' '삶' '석' '섭'
'셈' '송' '순' '숲' '쉬' '쉽' '슨' '슴' '습' '슷' '싫' '싶' '싸' '쌀' '써' '씀' '씬' '씻' '앉' '않' '앙'
'앞' '애' '억' '얻' '엇' '옆' '옛' '옳' '옷' '완' '왜' '욱' '육' '을' '읽' '잃' '잇' '잊' '잔' '잡' '절'
'젊' '접' '좀' '즉' '즘' '진' '짓' '징' '짧' '째' '쩌' '쪽' '찬' '찮' '찰' '참' '채' '처' '천' '첫' '청'
'초' '층' '친' '칠' '침' '컴' '코' '크' '큼' '탕' '텔' '투' '팀' '퍼' '평' '풀' '퓨' '함' '항' '혀' '호'
'혹' '황' '획' '효' '훌' '훨' '흐' '흔'

REFERENCES

[1] MacStories, "Online vs. offline handwriting recognition," 2016, [Online; accessed April 04, 2020]. [Online]. Available: https://www.macstories.net/reviews/nebos-handwriting-recognition-elevates-your-notes/

[2] Wikipedia, the free encyclopedia, "Center of excellence for document analysis and recognition," 2018, [Online; accessed April 04, 2020]. [Online]. Available: https://en.wikipedia.org/wiki/Center_of_Excellence_for_Document_Analysis_and_Recognition

[3] CBDAR, "Summarizing lecture videos by key handwritten content regions," 2018, [Online; accessed April 04, 2020]. [Online]. Available: https://cbdar2019.univ-lr.fr/wp-content/uploads/2019/10/CBDAR2019_Oral_2_a_01_CBDAR_bhargava.pdf

[4] B. Ahn, J. Ryu, H. I. Koo, and N. I. Cho, "Textline detection in degraded historical document images," *EURASIP Journal on Image and Video Processing*, vol. 2017, no. 1, p. 82, 2017.

[5] K. M. Sayre, "Machine recognition of handwritten words: A project report," *Pattern recognition*, vol. 5, no. 3, pp. 213–228, 1973.

[6] Margaret Rouse, "What is an Artificial Neural Network (ANN)," 2019, [Online; accessed April 04, 2020]. [Online]. Available: https://searchenterpriseai.techtarget.com/definition/neural-network

[7] Keith Enevoldsen, thinkzone.wlonk.com, "Korean alphabet (Hangeul)," 2016, [Online; accessed April 04, 2020]. [Online]. Available: https://thinkzone.wlonk.com/Language/Korean.htm

[8] U. Pal and S. Datta, "Segmentation of bangla unconstrained handwritten text," in *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.* Citeseer, 2003, pp. 1128–1132.

[9] U. Pal, K. Roy, and F. Kimura, "A lexicon driven method for unconstrained Bangla handwritten word recognition," in *Tenth International Workshop on Frontiers in Handwriting Recognition*, 2006.

[10] ——, "A lexicon-driven handwritten city-name recognition scheme for Indian postal automation," *IEICE transactions on information and systems*, vol. 92, no. 5, pp. 1146–1158, 2009.

[11] S. Basu, R. Sarkar, N. Das, M. Kundu, M. Nasipuri, and D. K. Basu, "A fuzzy technique for segmentation of handwritten Bangla word images," in *2007 International Conference on Computing: Theory and Applications (ICCTA'07)*. IEEE, 2007, pp. 427–433.

[12] R. Sarkar, S. Malakar, N. Das, S. Basu, M. Kundu, and M. Nasipuri, "Word extraction and character segmentation from text lines of unconstrained handwritten Bangla document images," *Journal of Intelligent Systems*, vol. 20, no. 3, pp. 227–260, 2011.

[13] R. Sarkar, N. Das, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu, "A two-stage approach for segmentation of handwritten Bangla word images," in *Proceedings of International Conference on Frontiers in Handwriting Recognitions*. Citeseer, 2008, pp. 403–408.

[14] A. Roy, T. K. Bhowmik, S. K. Parui, and U. Roy, "A novel approach to skew detection and character segmentation for handwritten Bangla words," in *Digital Image Computing: Techniques and Applications (DICTA'05)*. IEEE, 2005, pp. 30–30.

[15] T. Bhowmik, A. Roy, and U. Roy, "Character segmentation for handwritten Bangla words using artificial neural network," *Proc. 1st IAPR TC3 NNLDAR*, 2005.

[16] S. Bag, P. Bhowmick, G. Harit, and A. Biswas, "Character segmentation of handwritten Bangla text by vertex characterization of isothetic covers," in *2011 Third National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics*. IEEE, 2011, pp. 21–24.

[17] A. Bishnu and B. Chaudhuri, "Segmentation of Bangla handwritten text into characters by recursive contour following," in *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR'99 (Cat. No. PR00318)*. IEEE, 1999, pp. 402–405.

[18] K. Bhattacharyya and K. K. Sarma, "ANN-based innovative segmentation method for handwritten text in Assamese," *arXiv preprint arXiv:0911.0907*, 2009.

[19] N. Tripathy and U. Pal, "Handwriting segmentation of unconstrained Oriya text," *Sadhana*, vol. 31, no. 6, pp. 755–769, 2006.

[20] N. K. Garg, L. Kaur, and M. Jindal, "Segmentation of handwritten Hindi text," *International Journal of Computer Applications*, vol. 1, no. 4, pp. 22–26, 2010.

[21] M. Hanmandlu, P. Agrawal, and B. Lall, "Segmentation of handwritten hindi text: A structural approach," *International Journal of Computer Processing of Languages*, vol. 22, no. 01, pp. 1–20, 2009.

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

[23] B. B. Chaudhuri and A. Kundu, "Synthesis of individual handwriting in bangla script," *Proceedings of the ICFHR*, 2008.

[24] U. Pal, T. Wakabayashi, and F. Kimura, "Handwritten Bangla compound character recognition using gradient feature," in *10th International Conference on Information Technology (ICIT 2007)*. IEEE, 2007, pp. 208–213.

[25] M. Mohamad, D. Nasien, H. Hassan, and H. Haron, "A review on feature extraction and feature selection for handwritten character recognition," *International Journal of Advanced Computer Science and Applications*, vol. 6, no. 2, pp. 204–212, 2015.

[26] T. K. Bhowmik, P. Ghanty, A. Roy, and S. K. Parui, "SVM-based hierarchical architectures for handwritten Bangla character recognition," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 12, no. 2, pp. 97–108, 2009.

[27] K. N. Reza and M. Khan, "Grouping of handwritten Bangla basic characters, numerals and vowel modifiers for multilayer classification," in *2012 International Conference on Frontiers in Handwriting Recognition*. IEEE, 2012, pp. 325–330.

[28] A. Majumdar and B. Chaudhuri, "Curvelet-based multi svm recognizer for offline handwritten bangla: a major indian script," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 1. IEEE, 2007, pp. 491–495.

[29] N. Das, B. Das, R. Sarkar, S. Basu, M. Kundu, and M. Nasipuri, "Handwritten Bangla basic and compound character recognition using MLP and SVM classifier," *arXiv preprint arXiv:1002.4040*, 2010.

[30] N. Das, K. Acharya, R. Sarkar, S. Basu, M. Kundu, and M. Nasipuri, "A novel GA-SVM based multistage approach for recognition of handwritten

Bangla compound characters," in *Proceedings of the International Conference on Information Systems Design and Intelligent Applications 2012 (INDIA 2012) held in Visakhapatnam, India, January 2012*. Springer, 2012, pp. 145–152.

[31] A. Das, A. K. Bhunia, P. P. Roy, and U. Pal, "Handwritten word spotting in Indic scripts using foreground and background information," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*. IEEE, 2015, pp. 426–430.

[32] N. Das, K. Acharya, R. Sarkar, S. Basu, M. Kundu, and M. Nasipuri, "A benchmark image database of isolated Bangla handwritten compound characters," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 17, no. 4, pp. 413–431, 2014.

[33] S. Mohiuddin, U. Bhattacharya, and S. K. Parui, "Unconstrained Bangla online handwriting recognition based on MLP and SVM," in *Proceedings of the 2011 Joint Workshop on Multilingual OCR and Analytics for Noisy Unstructured Text Data*. ACM, 2011, p. 16.

[34] S. Ahmed, M. R. Islam, and M. S. Azam, "Bangla hand written digit recognition using supervised locally linear embedding algorithm and support vector machine," in *2009 12th International Conference on Computers and Information Technology*. IEEE, 2009, pp. 390–393.

[35] Y. Wen, Y. Lu, and P. Shi, "Handwritten Bangla numeral recognition system and its application to postal automation," *Pattern recognition*, vol. 40, no. 1, pp. 99–107, 2007.

[36] S. Pal, V. Nguyen, M. Blumenstein, and U. Pal, "Off-line Bangla signature verification," in *2012 10th IAPR International Workshop on Document Analysis Systems*. IEEE, 2012, pp. 282–286.

[37] P. P. Roy, P. Dey, S. Roy, U. Pal, and F. Kimura, "A novel approach of Bangla handwritten text recognition using HMM," in *2014 14th International Conference on Frontiers in Handwriting Recognition*. IEEE, 2014, pp. 661–666.

[38] T. K. Bhowmik, U. Bhattacharya, and S. K. Parui, "Recognition of Bangla handwritten characters using an MLP classifier based on stroke features," in *International Conference on Neural Information Processing*. Springer, 2004, pp. 814–819.

[39] A. Dutta and S. Chaudhury, "Bengali alpha-numeric character recognition using curvature features," *Pattern Recognition*, vol. 26, no. 12, pp. 1757–1770, 1993.

[40] S. Bag, P. Bhowmick, and G. Harit, "Recognition of Bengali handwritten characters using skeletal convexity and dynamic programming," in *2011 Second International Conference on Emerging Applications of Information Technology*. IEEE, 2011, pp. 265–268.

[41] S. Arora, D. Bhattacharjee, M. Nasipuri, D. K. Basu, and M. Kundu, "Combining multiple feature extraction techniques for handwritten Devnagari character recognition," in *2008 IEEE Region 10 and the Third international Conference on Industrial and Information Systems*. IEEE, 2008, pp. 1–6.

[42] N. Sharma, U. Pal, F. Kimura, and S. Pal, "Recognition of off-line handwritten devnagari characters using quadratic classifier," in *Computer Vision, Graphics and Image Processing*. Springer, 2006, pp. 805–816.

[43] D. Sharma and P. Jhajj, "Recognition of isolated handwritten characters in Gurmukhi script," *International Journal of Computer Applications*, vol. 4, no. 8, pp. 9–17, 2010.

[44] M. Kumar, M. Jindal, and R. Sharma, "K-nearest neighbor based offline handwritten Gurmukhi character recognition," in *2011 International Conference on Image Information Processing*. IEEE, 2011, pp. 1–4.

[45] E. Hassan, S. Chaudhury, M. Gopal, and J. Dholakia, "Use of MKL as symbol classifier for Gujarati character recognition," in *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*. ACM, 2010, pp. 255–262.

[46] M. A. R. Alif, S. Ahmed, and M. A. Hasan, "Isolated Bangla handwritten character recognition with convolutional neural network," in *Computer and Information Technology (ICCIT), 2017 20th International Conference of*, 2017, pp. 1–6.

[47] K. Kowsari, M. Heidarysafa, D. E. Brown, K. J. Meimandi, and L. E. Barnes, "RMDL: Random multimodel deep learning for classification," in *Proceedings of the 2nd International Conference on Information System and Data Mining*, 2018, pp. 19–28.

[48] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Convolutional neural network committees for handwritten character classification," in *2011 International Conference on Document Analysis and Recognition*. IEEE, 2011, pp. 1135–1139.

[49] V. V. Romanuke, "Training data expansion and boosting of convolutional neural networks for reducing the MNIST dataset error rate," *Research Bulletin of the National Technical University of Ukraine*, vol. 0, no. 6, pp. 29–34, 2016.

[50] K. Dutta, P. Krishnan, M. Mathew, and C. Jawahar, "Offline handwriting recognition on Devanagari using a new benchmark dataset," in *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. IEEE, 2018, pp. 25–30.

[51] "Handwritten Hangul Datasets: PE92, SERI95, and HanDB." https://github.com/callee2006/HangulDB, 1992.

[52] G.-R. Park, I.-J. Kim, and C.-L. Liu, "An evaluation of statistical methods in handwritten Hangul recognition," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 16, no. 3, pp. 273–283, 2013.

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

[54] L. Rothacker, M. Rusinol, and G. A. Fink, "Bag-of-features HMMs for segmentation-free word spotting in handwritten documents," in *2013 12th International Conference on Document Analysis and Recognition*. IEEE, 2013, pp. 1305–1309.

[55] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[56] S. Lloyd, "Least squares quantization in PCM," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[57] S. Sudholt, L. Rothacker, and G. A. Fink, "Learning local image descriptors for word spotting," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2015, pp. 651–655.

[58] L. Rothacker, S. Vajda, and G. A. Fink, "Bag-of-features representations for offline handwriting recognition applied to Arabic script," in *2012 International Conference on Frontiers in Handwriting Recognition*. IEEE, 2012, pp. 149–154.

[59] X. Zhang, U. Pal, and C. L. Tan, "Segmentation-free Keyword spotting for Bangla handwritten documents," in *2014 14th International Conference on Frontiers in Handwriting Recognition*. IEEE, 2014, pp. 381–386.

[60] S. Wshah, G. Kumar, and V. Govindaraju, "Script independent word spotting in offline handwritten documents based on hidden markov models," in *2012 International Conference on Frontiers in Handwriting Recognition*. IEEE, 2012, pp. 14–19.

[61] A. K. Bhunia, A. Das, P. P. Roy, and U. Pal, "A comparative study of features for handwritten Bangla text recognition," in *2015 13th international conference on document analysis and recognition (ICDAR)*. IEEE, 2015, pp. 636–640.

[62] R. Shekhar and C. Jawahar, "Word image retrieval using bag of visual words," in *2012 10th IAPR International Workshop on Document Analysis Systems*. IEEE, 2012, pp. 297–301.

[63] S. Vajda and A. Belaïd, "Structural information implant in a context based segmentation-free HMM handwritten word recognition system for latin and Bangla script," in *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*. IEEE, 2005, pp. 1126–1130.

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

[65] S. Bhowmik, M. G. Roushan, R. Sarkar, M. Nasipuri, S. Polley, and S. Malakar, "Handwritten Bangla word recognition using HOG descriptor," in *2014 Fourth International Conference of Emerging Applications of Information Technology*. IEEE, 2014, pp. 193–197.

[66] T. K. Bhowmik, S. K. Parui, and U. Roy, "Discriminative HMM training with GA for handwritten word recognition," in *2008 19th International Conference on Pattern Recognition*. IEEE, 2008, pp. 1–4.

[67] C. Adak, B. B. Chaudhuri, and M. Blumenstein, "Offline cursive Bengali word recognition using CNNs with a recurrent model," in *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 2016, pp. 429–434.

[68] A. Ul-Hasan and T. M. Breuel, "Can we build language-independent OCR using LSTM networks?" in *Proceedings of the 4th International Workshop on Multilingual OCR*. ACM, 2013, p. 9.

[69] T. Bluche, H. Ney, and C. Kermorvant, "A comparison of sequence-trained deep neural networks and recurrent neural networks optical modeling for handwriting recognition," in *International Conference on Statistical Language and Speech Processing*. Springer, 2014, pp. 199–210.

[70] F. Menasri, J. Louradour, A.-L. Bianne-Bernard, and C. Kermorvant, "The A2iA French handwriting recognition system at the Rimes-ICDAR2011 competition," in *Document Recognition and Retrieval XIX*, vol. 8297. International Society for Optics and Photonics, 2012, p. 82970Y.

[71] F. Stahlberg and S. Vogel, "The QCRI recognition system for handwritten Arabic," in *International Conference on Image Analysis and Processing*. Springer, 2015, pp. 276–286.

[72] K. Dutta, P. Krishnan, M. Mathew, and C. Jawahar, "Towards accurate handwritten word recognition for Hindi and Bangla," in *National Conference on Computer Vision, Pattern Recognition, Image Processing, and Graphics*. Springer, 2017, pp. 470–480.

[73] "Handwritten Bangla numeral recognition using deep long short term memory, author=Ahmed, Mahtab and Akhand, MAH and Rahman, MM Hafizur," in *2016 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M)*. IEEE, 2016, pp. 310–315.

[74] V. Chavan, A. Malage, K. Mehrotra, and M. K. Gupta, "Printed text recognition using BLSTM and MDLSTM for Indian languages," in *2017 Fourth International Conference on Image Information Processing (ICIIP)*. IEEE, 2017, pp. 1–6.

[75] B. Chakraborty, P. S. Mukherjee, and U. Bhattacharya, "Bangla online handwriting recognition using recurrent neural network architecture," in *Proceedings of the tenth Indian conference on computer vision, graphics and image processing*, 2016, pp. 1–8.

[76] T. Karayil, A. Ul-Hasan, and T. M. Breuel, "A segmentation-free approach for printed Devanagari script recognition," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2015, pp. 946–950.

[77] N. Sankaran and C. Jawahar, "Recognition of printed Devanagari text using BLSTM Neural Network," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. IEEE, 2012, pp. 322–325.

[78] "CMATERdb: The pattern recognition database repository," http://code.google.com/p/cmaterdb, March 2018.

[79] R. Sarkar, N. Das, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu, "CMATERdb1: a database of unconstrained handwritten Bangla and Bangla–English mixed script document image," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 15, no. 1, pp. 71–83, Feb 2011.

[80] N. Das, K. Acharya, R. Sarkar, S. Basu, M. Kundu, and M. Nasipuri, "A benchmark image database of isolated Bangla handwritten compound characters," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 17, no. 4, pp. 413–431, May 2014.

[81] N. Das, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu, "A genetic algorithm based region sampling for selection of local features in handwritten digit recognition application," *Applied Soft Computing*, vol. 12, no. 5, pp. 1592–1606, 2012.

[82] N. Das, S. Basu, R. Sarkar, M. Kundu, M. Nasipuri *et al.*, "An improved feature descriptor for recognition of handwritten Bangla alphabet," in *Proceedings of 2nd International Conference on Signal and Image Processing (ICSIP)*, 2009, pp. 451–454.

[83] U. Bhattacharya, M. Shridhar, S. K. Parui, P. K. Sen, and B. B. Chaudhuri, "Offline recognition of handwritten Bangla characters: an efficient two-stage approach," *Pattern Analysis and Applications*, vol. 15, no. 4, pp. 445–458, June 2012.

[84] M. Biswas, R. Islam, G. K. Shom, M. Shopon, N. Mohammed, S. Momen, and A. Abedin, "Banglalekha-isolated: A multi-purpose comprehensive dataset of handwritten Bangla isolated characters," *Data in brief*, vol. 12, pp. 103–107, 2017.

[85] N. Majid and E. H. Barney Smith, "Segmentation-free Bangla offline handwriting recognition using sequential detection of characters and diacritics with a Faster R-CNN," in *International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2019.

[86] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 6, pp. 1137–1149, 2017.

[87] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[88] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "ImageNET large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[89] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[90] K. Murphy, "Machine learning: a probabilistic approach," *Massachusetts Institute of Technology*, pp. 1–21, 2012.

[91] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8, 1966, pp. 707–710.

[92] N. Majid and E. H. Barney Smith, "Boise State Bangla Handwriting Dataset," https://doi.org/10.18122/saipl/1/boisestate, 2018.

[93] N. Majid and E. H. Barney Smith, "Introducing the Boise State Bangla Handwriting dataset and an efficient offline recognizer of isolated Bangla characters," in *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 2018, pp. 380–385.

[94] T. Y. Kong and A. Rosenfeld, *Topological algorithms for digital image processing*. Elsevier, 1996, vol. 19.

[95] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[96] K. Zagoris, I. Pratikakis, A. Antonacopoulos, B. Gatos, and N. Papamarkos, "Distinction between handwritten and machine-printed text based on the bag of visual words model," *Pattern Recognition*, vol. 47, no. 3, pp. 1051–1062, 2014.

[97] S. Mukherjee, P. Kumar, and P. P. Roy, "Fusion of spatio-temporal information for Indic word recognition combining online and offline text data," *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, vol. 19, no. 2, pp. 1–24, 2019.

[98] C. Clausner, A. Antonacopoulos, T. Derrick, and S. Pletschacher, "Icdar2019 competition on recognition of early indian printed documents–reid2019," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2019, pp. 1527–1532.

[99] Y. Xu and G. Nagy, "Prototype extraction and adaptive OCR," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 12, pp. 1280–1296, 1999.

[100] D. Lopresti and G. Nagy, "Tools for monitoring, visualizing, and refining collections of noisy documents," in *Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data*, 2009, pp. 9–16.

[101] A. Roy, N. Das, R. Sarkar, S. Basu, M. Kundu, and M. Nasipuri, "Region selection in handwritten character recognition using artificial bee colony optimization," in *Emerging Applications of Information Technology (EAIT), 2012 Third International Conference on*. IEEE, 2012, pp. 183–186.

[102] A. F. R. Rahman, R. Rahman, and M. C. Fairhurst, "Recognition of handwritten Bengali characters: a novel multistage approach," *Pattern Recognition*, vol. 35, no. 5, pp. 997–1006, 2002.

[103] T. K. Bhowmik, P. Ghanty, A. Roy, and S. K. Parui, "SVM-based hierarchical architectures for handwritten Bangla character recognition," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 12, no. 2, pp. 97–108, March 2009.

[104] U. Bhattacharya, S. Parui, and B. Shaw, "A hybrid scheme for recognition of handwritten Bangla basic characters based on HMM and MLP classifiers," in *Advances In Pattern Recognition*. World Scientific, 2007, pp. 101–106.

[105] N. Majid and E. H. B. Smith, "Performance comparison of scanner and camera-acquired data for Bangla offline handwriting recognition," in *2019 Workshop on Camera-based Document Recognition and Retrieval (CBDAR)*, vol. 4. IEEE, 2019, pp. 31–36.

[106] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

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

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

[109] S. Mori, K. Yamamoto, H. Yamada, and T. Saito, "On a handprinted Kyoiku-Kanji character data base," *Bull. Electrotech. Lab*, vol. 43, no. 11-12, pp. 752–773, 1979.

[110] T. SAITO, "On the data base ETK9B of handprinted characters in JIS Chinese characters and its analysis," *IEICE trans*, vol. 68, no. 4, pp. 757–772, 1985.

[111] N. Majid and E. H. Barney Smith, "Boise State Bangla Handwriting Dataset," https://doi.org/10.18122/saipl/1/boisestate, 2018.

[112] "1000 Most Common Korean Words." https://1000mostcommonwords.com/1000-most-common-korean-words/.