Segmentation of Bangla Compound Characters: Underlying Simple Character Detection from Handwritten Compound Characters

by

Md Raihanul Islam Bhuiyan
23341083
Mahin Shahriar Efaz
23341084
Tanjim Reza
20101065
Aditi Saha Ria
23341085

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

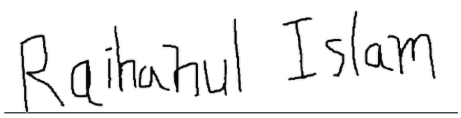
Department of Computer Science and Engineering
School of Data and Sciences
Brac University
September 2023

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

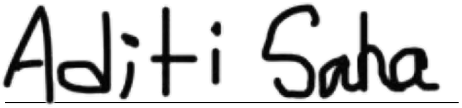
Student's Full Name & Signature:



Md Raihanul Islam Bhuiyan
23341083

Mahin Shahriar Efaz
23341084

Tanjim Reza
20101065



Aditi Saha Ria
23341085

Approval

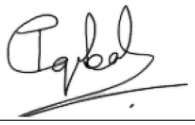
The thesis titled "Segmentation of Bangla Compound Characters: Underlying Simple Character Detection from Handwritten Compound Characters" submitted by

1. Md Raihanul Islam Bhuiyan (23341083)
2. Mahin Shahriar Efaz (23341084)
3. Tanjim Reza (20101065)
4. Aditi Saha Ria (23341085)

Of Summer, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on September 2023.

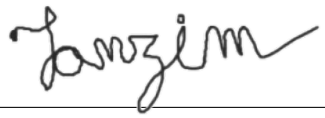
Examining Committee:

Supervisor:
(Member)



Dr. Muhammad Iqbal Hossain
Associate Professor
Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)



Mr. Md. Tanzim Reza
Lecturer
Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

We hereby declare that this thesis is based on our own fndings from our own research discoveries. All other sources used in this work have been properly acknowledged. Furthermore, we confrm that this thesis has not been submitted or presented, either in its entirety or partially for the purpose of receiving a degree from any other educational institution or university.

Abstract

Bangla is one of the most popular languages in the world and more than 210 Million people use it as their first or second language. The literature of Bangla has a rich history and dates back thousands of years. However, Bangla characters have a compound structure; some contain more than one simple character to form a single compound character. There is a lot of work on character recognition but the structure of the compound characters makes the detection of Bangla Compound Characters a difficult task. The existing method on Bangla compound characters uses a list of compound characters as the dataset, trains models on the whole image, and detects the characters. Using this method on handwritten characters, the accuracy decreases when the characters are slightly different from the train images or the characters consist of two different simple characters that are not in the train images. To overcome this problem, our research focus is to detect character type i.e. simple or compound using VGG 16 architecture and YOLO, and if it is a compound character, it can detect the underlying simple characters inside the compound characters. To conduct our research, we created a new Bengali Handwritten character dataset called **"BanglaBorno"** as the existing datasets had some limitations in the quantity of compound characters or the quality of the images.

Keywords: Bangla Text Recognition, Compound Characters, VGG16 architecture, YOLOv8 models, BanglaBorno.

Dedication

We want to dedicate all of our sacrifces and educational eforts to our great parents, without whom we would be worthless. We also dedicate our thesis to Dr. Muhammad Iqbal Hossain sir, who served as our supervisor, guided us, and showed us how to develop our skills and personalities as successful professionals.

Acknowledgement

Firstly, all praise to the God for whom our thesis have been completed without any major interruption.
Secondly, to our Supervisor Dr. Muhammad Iqbal Hossain sir for his kind support and advice in our work. He helped us whenever we needed help.
And finally to our parents without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Handwritten character recognition has emerged as a prominent interest due to its broad range of applications [1]. For thousands of years, writing by hand has been the most convenient method of storing information. However, digital copies of handwritten images are also frequently used to store information in the twenty-first century. Recognizing handwritten forms, understanding different fonts used in posters and newspapers, translating languages, and many other fields all require the ability to create digital copies of handwritten images.

Recognizing handwritten characters is a very fragile technique as the fonts and sizes of each character vary depending on the person's handwriting style. So it is very difficult to get a 100% accuracy result [2].

The number of works that have been done on English handwritten character recognition and other similar languages is very overwhelming, however, the number is very unpleasant in Bangla. Even though Bangla is the seventh most spoken language in the world it is a very disappointing fact that there has not been enough research done on recognizing Bangla handwritten characters [3]. Bangla has very enriched characters and some characters are a combination of two or more characters. This makes the task of recognizing handwritten Bangla characters difficult. Convolutional Neural Network (CNN), Support Vector Machine (SVM), Deep neural network, Deep learning, Machine learning, Multilayer Perceptron (MLP), Modified Quadratic Discriminant Function (MQDF), a combination of deep convolutional neural network with Bidirectional long short-term memory (CNN-BiLSTM) and many more methods were used to detect handwritten Bangla characters.

Furthermore, all the research that has been done on detecting handwritten Bangla compound characters, detects it by using the whole image. This means that they consider each compound character as a whole new character. But none of the Bengali handwritten character datasets contain all the compound characters in them as the combinations of simple characters to form up a compound character has no fixed amount and as a result, there have not been any fixed numbers of compound characters yet. However, when we detect which simple characters are in a compound character, it is not a very complex task to predict the compound character inside it. So we came up with the approach to detect the underlying simple characters in the compound character so that it can work for every compound character even if it is not included in the training dataset.

In this paper, we have tried to overcome the limitations of Bangla handwritten basic and compound character detection and detect both simple and compound Bangla

handwritten characters. We have also attempted to detect the underlying simple characters in a compound character.

1.1 Research Problem

English, Chinese, and some other languages have a very remarkable number of researches on recognizing handwritten characters and the success rates are outstanding. This successful research on English and other languages have made text-to-speech and many more things related to text detection possible. On the other hand, the Bangla language is still not there in the text recognition field like English and other languages because of its obstacles in detecting complex characters.
The key problem in recognizing handwritten Bangla characters is the compound characters because of its cursive nature [4]. Again some compound characters are almost identical to each other which makes the process very difficult. These types of characters can only be recognized by short straight lines, circular curves etc. [5]. The compound characters become more complex to recognize by the increasing number of basic character combinations [6].
Furthermore, some handwritten characters do not resemble their printed characters and this also makes the recognition process tough [6]. Also, each person has a unique style of writing, and the shape of the characters becomes different. This means that handwritten characters add a lot of variability in the characters which makes recognition problems challenging [7].
Though there is some research on detecting Bangla handwritten compound characters, however, no research has been done on recognizing the simple characters by which the compound character is formed. All the research done until now has only detected the compound characters as the whole character. According to Das (2014) [8] Bangla script has more than 300 compound characters and none of the existing datasets contain all of the compound characters. This creates a challenge for the existing models to identify a compound character that is not included in the training dataset. So it is one of the significant problems of this research that no work of segmentation for Bangla compound characters has been done yet.
To eliminate this problem we took a new approach of segmenting the compound character and identifying the underlying simple character. This method will help the model to detect a compound character that is not present in the dataset. For instance, if a compound character "ক্ল" does not exist in a training dataset but the dataset has "ক্ষ" and "ম্ন". We aim to train our model by segmenting the compound character so that it can detect the "ক" and "ল" from "ক্ল" by using its knowledge from "ক্ষ" and "ম্ন". By using segmentation, we can overcome the disadvantage of not having all the compound characters in the training set.
Keeping these problems in mind, the question implies if it is possible to separate the underlying simple characters from the handwritten Bangla compound characters accurately and also will it be able to segment the compound characters into simple characters precisely?
In this research, we aim to answer the above question by studying and exploring the character recognition field.

1.2 Research Objectives

The research aims to develop a system that can detect Bangla Compound characters with better accuracy. The objectives of this research are:

- To improve the current state of Bangla OCR using segmentation techniques
- To create a new Bengali Handwritten Character Dataset **BanglaBorno** with a more enriched amount of compound characters.
- To deeply understand the VGG-16, and YOLO Models and use them in Bangla OCR
- To detect Bangla Compound characters including simple characters
- Detect characters from handwritten texts with different handwriting styles
- Detect the underlying simple characters from the compound characters accurately

1.3 Thesis Structure

We have followed the following structure for each step in our research

- Chapter 1: Handwritten Character recognition has lots of applications in various fields and there are lots of work on Bangla Compound character recognition but there are so many constraints and no work on segmentation has been done yet.
- Chapter 2: In this section, we have explored and analyzed all the existing work on Bangla handwritten character recognition and understood their shortcomings and solutions. We have explored different models that we will be using in our research to find out the ways of optimizing our test and training sets.
- Chapter 3: In this section, we have explored all the available datasets on Bangla characters and analyzed them to find out the limitations and lastly proposed our own dataset **BanglaBorno**.
- Chapter 4: In this chapter, we have explored different models that we will be using in our research to find out the ways of optimizing our test and training sets. We have processed our dataset in various ways to make it suitable for training. We have used augmentation, thresholding, annotation, and other techniques. After that, we trained and tested the models.
- Chapter 5: In this section, we have analyzed our results, compared the accuracy between different models, and found out our limitations and improvements.
- Chapter 6: Finally we have concluded our works and explained the future works that can be done further in this research.

Chapter 2

Literature Review

2.1 Existing Works

At present, there is a lot of work going on to detect handwritten characters from images. For simple languages like English, we have achieved great accuracy in detecting the language. However, there have been very few works on the Bangla language for handwriting recognition. The most difficult part about detecting the handwritten Bangla language is the compound characters in it. Compound characters are made of two or more characters together and it is hard for the computer to recognize these characters.

The technique that is used to transform a picture of a text character into a text format that can be used for data processing is known as optical character recognition (OCR). Using this process, there exists a great deal of work for character recognition. However, when it comes to bad-quality images of text characters, the accuracy falls as it becomes difficult to understand the text. Therefore, in a work [9], the authors introduced a technique to increase the accuracy of the recognition using a Deep Neural Network. They implemented a deep neural network, which is a pre-trained V3 model with the application of the Transfer learning concept. Transfer learning accelerates the training speed and improves character recognition precision. The rate of inaccuracy in picture text recognition was lowered by around 21.5% with the introduction of this improved OCR.

In another research, the study demonstrates the handwritten Telugu compound character Guninthalu (Muppalaneni, 2020) [10]. In their Convolutional, Pooling and Fully connected layer approach they got 79.61% test accuracy and 96.13% training accuracy. They have created a Convolutional Neural Network-based machine learning model for Telugu Handwritten Gunithalu. Even though they have gained high accuracy, their model could not differentiate between two compound characters. The letter samples (Kru) wass projected as (Ku) and (Ko) are predicted as (Ku) (Kaa). They have experimented with several structures to improve accuracy, but it was difficult to improve accuracy with a tiny dataset because each letter was similar to the others.

In an additional research on Marathi Compound Characters, a unique segmentation method for handwritten Marathi compound letters is proposed (Golait & Malik, 2016) [11]. The proposed method used the idea of Minutiae extraction from fingerprints to split apart complex characters. Basically, Morphological processes, such as erosion and dilation, serve to create the segments. To separate a single character

from a string of compound ones, we look for ends and forks. The proposed method additionally made use of the hit-or-miss transform morphological operation to pinpoint the endpoint and bifurcation point. Based on the experimental findings, we can confidently pinpoint termination and bifurcation points with a 90% degree of accuracy.

In a research, Deep Convolutional Neural Network is used on all the Bengali characters (Purkaystha et al, 2017) [12]. It was applied on the BanglaLekha-Isolated dataset. They have achieved 98.66% accuracy on Bangla digits, 94.99% on Bangla vowels, 91.60% accuracy on compound letters, 91.23% accuracy on Bangla alphabets. The average accuracy of the Bangla characters is 89.93% in their research. The accuracy of compound characters is very low compared to other works. Again here the author mentioned that their models make some errors in generating the output different from their label.

In another work, MLP and SVM classifiers were used to recognize Bangla basic and compound handwritten characters (Das et al, 2010) [13]. Here, the authors have used shadow features, longest run features and quad-tree based features to recognize the characters. In this research, authors have gained 79.25% accuracy using MLP and 80.510% accuracy using SVM. In the paper, the author showed a figure of misclassified characters by the approached method.

In a paper, Hasan et al. (2019) [14] have approached with deep convolutional neural network combined with Bi-directional short-term memory. They combined these two methods to identify the dependencies in images and they believed that this would give a better accuracy rate than the other models. CMATERDB 3.1.3.3 dataset was used in the paper and to improve the accuracy the authors used a preprocessing method on the dataset. The proposed method gave 98.50% test accuracy. However, the hybrid model sometimes has a hard time recognizing the similarly structured characters and misclassifies them.

In another research, Kibria et al. (2020) [15] used the SVM classifier for handwritten character detection and recognition. Here, they utilized CMATERdb 3.1.3.3 for the dataset, which consists of 55,278 Bangla compound character gray-level images grouped into 199 pattern shapes and 171 classes. Longest Run Feature based on CG based Partitioning, Histogram of oriented gradients (HOG) feature and Diagonal Feature, are the three feature sets which were integrated to estimate the recognition performance on the chosen dataset. Their proposed approach was able to obtain 89.43% precision, 88.73% recall and 88.89% F1 score.

In a work, Roy et al (2017) [7] applied a revolutionary deep learning methodology and contrasted with the traditional methodologies of Support Vector Machines (SVM) and Deep Convolutional Neural Nets (DCNNs). The authors performed the training to DCNN layer-by-layer. The procedure was implemented on the CMATERdb 3.1.3.3 dataset. Using the 8520 test samples of the dataset and a total of 34439 training samples, which are image samples, they achieved recognition accuracy of 90.33%. Furthermore, in order to improve the convergence rate, they applied the RMSProp algorithm, which produced successful results. The proposed method misclassifies the highly structured compound characters which have at least one similar basic character in it.

In another research by Fardous & Afroge (2019) [16], CMATERdb 3.1.3.3 dataset was used. The authors have used the Convolutional Neural network Model using convolutional layers and fully connected layers. This method has gained an average

accuracy of 95.5%. However, it's unfortunate that the method fails to classify some characters and misrecognizes them as other characters.
In another research by Pal et al (2007) [17] worked on Bangla compound characters using Modified Quadratic Discriminant Function (MQDF). Their research uses the information of the direction that they get from the arc tangent of the gradient. The authors have used gray level images and applied 2x2 filtering 4 times on that. A non linear size normalization and Roberts filter is used on the images. Arc tangent of the gradient is converted into 32 directions and its strength gradient is calculated in each direction. They have used 20,543 samples and gained 85.90% accuracy using 5-fold cross validation technique. The drawback of this research is that the proposed method cannot recognize the actual printed characters of the handwritten characters. It generates some similar but not the same compound printed characters which is actually an error.
The authors [18] have used DCNN with Dropout and ELU in the CMATERdb 3.1.3.3 dataset and gained an accuracy of 93.68%. But even though the accuracy is large, in some cases the algorithm misclassifies in the testing phase. The authors think that the reason behind this is the small size of the dataset.

2.2 Model Architectures

As there have been few studies on the recognition of Bangla compound characters, optical character recognition (OCR) continues to be a difficult task. Deep learning has proven to be a highly effective technology due to its ability to process vast quantities of data. Therefore, we employed a number of deep learning approaches and models on a dataset to determine which model provides the best results.

2.2.1 VGG-16 Architecture

VGG16, also known as Visual Geometry Group 16, a convolutional neural network, is named due to its architecture which consists of 16 layers proposed by K. Simonyan and A. Zisserman from Oxford University. So the 16 in VGG16 indicates its 16 layers architecture. This 16 layered model contains 138 million parameters which is a very large model compared to today's standard [19]. The main attraction of the VGGNet16 architecture lies in its inherent simplicity. Among the 16 layers of VGG16 architecture 13 layers are convolutional layers and the other remaining layers are fully connected layers.

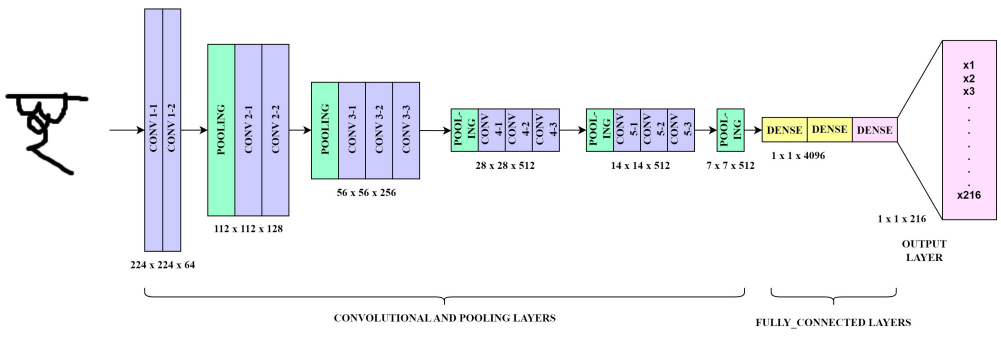


Figure 2.1: VGG-16 Model Architecture

VGG16 architecture is used widely for image processing, recognition and classification work. The VGG16 architecture is designed to process image inputs with dimensions of 224×224. The convolutional layers of VGG16 architecture use convolutional filters of the size of 3×3, which is the smallest possible size. To linearly transform the input the VGG architecture also incorporates a 1×1 convolutional filter. This makes sure that the feature maps of spatial dimensions do not change following each convolutional layer. The hidden layers increase both training time and memory usage while providing only marginal enhancements to the overall accuracy. Furthermore, a pooling layer following several convolutional layers helps to reduce the spatial dimensionality and the number of parameters of the feature maps created by each convolution step while making sure of getting the maximum salient information.

2.2.2 YOLOv8 Architecture

YOLO, You Only Look Once is a very fast, accurate but simple object detection model developed in 2015 by Joseph Redmon of the University of Washington. In Yolo, A single convolutional network is responsible for continuously predicting multiple bounding boxes and class probabilities for those boxes. YOLO resizes the input images to 448x448 and then runs a single convolutional network on the image and finally thresholds the detections by the model's confidence. Yolo is very fast, the base network runs at 45 frames per second which makes it perfect for processing real-time streaming video with less than 25 milliseconds of latency and more than twice the mean average precision of other real-time systems.
Unlike other detection systems that use classifiers to perform detection, YOLO considers object detection as a single regression problem and as the name suggests You Only Look Once to predict the objects from an image which makes it fast. The Yolo algorithm mainly works on three techniques- 1. Grid Blocks, 2. Bounding Box Regression, 3. Intersection Over Union (IOU).
First, the image is divided into grids, 7x7 in the original YOLO Algorithm. The grid cells are of equal dimensions and every grid cell will detect objects that appear within them. The object's center-appearing cell is responsible for detecting it on that particular cell.
Secondly, For each cell in the image, YOLO predicts bounding box coordinates that highlight an object in an image and the confidence scores.
The confidence score tells the possibility of having an object in that cell and the accuracy of detection. YOLO only chooses one predicted bounding box using the confidence scores and predicts conditional probabilities for each class.
Finally, The first step of filtering is done through the Intersection over the Union threshold. In YOLO, the filtering of box predictions by confidence level is done manually with a default 0.5 threshold value.
IoU calculates the overlap of the two bounding boxes divided by their union to provide an accurate estimate. The IoU value ranges from 0 to 1 where 0 means no overlap between boxes and 1 represents complete overlap. It is essential to determine if the predicted object is a true positive or a false positive.
The most recent version of YOLO was released by Ultralytics in January 2023. YOLOv8 provides five different scaled versions from YOLOv8n (nano) to YOLOv8x (extra-large), and supports object detection, segmentation, pose estimation, track-

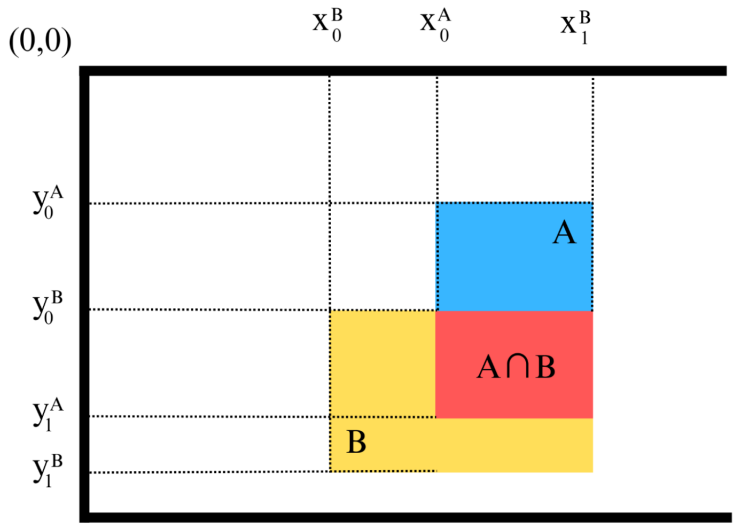


Figure 2.2: Intersection over union

ing, and classification. YOLOv8's architecture is similar to YOLOv5 but more advanced. Some changes are made in the characteristics of CSPLayer, which is called the C2f module in v8, to combine high-level features with contextual information for improved detection accuracy. It employs an anchor-free model with a decoupled head, enabling independent processing of objectness, classification, and regression tasks for enhanced accuracy while the output layer uses the sigmoid function for objectness score (detection probability) and softmax for class probabilities. For bounding box loss YOLOv8 uses CIoU and DFL loss functions and for classification loss, it uses binary cross-entropy, which provides improved object detection, especially for smaller objects. Additionally, YOLOv8 offers a semantic segmentation model called YOLOv8-Seg, which uses CSPDarknet53 feature extractor and C2f module, which achieves state-of-the-art results for object detection and semantic segmentation, maintaining high speed and efficiency [20].

Chapter 3

Dataset Description

3.1 Existing Datasets

3.1.1 CMATERdb 3.1.3.3 Dataset

The CMATERdb 3.1.3.3 dataset [21] is a comprehensive collection of handwritten characters commonly used in the Bengali script created by the Center for Microprocessor Application for Training Education and Research, Computer Science and Engineering Department, Jadavpur University, Kolkata, India. It is composed of a vast collection of 55,279 images and encompasses a total of 171 unique classes. The dataset displays a range of diverse characteristics that make it suitable for the creation and evaluation of models related to the recognition of compound characters.

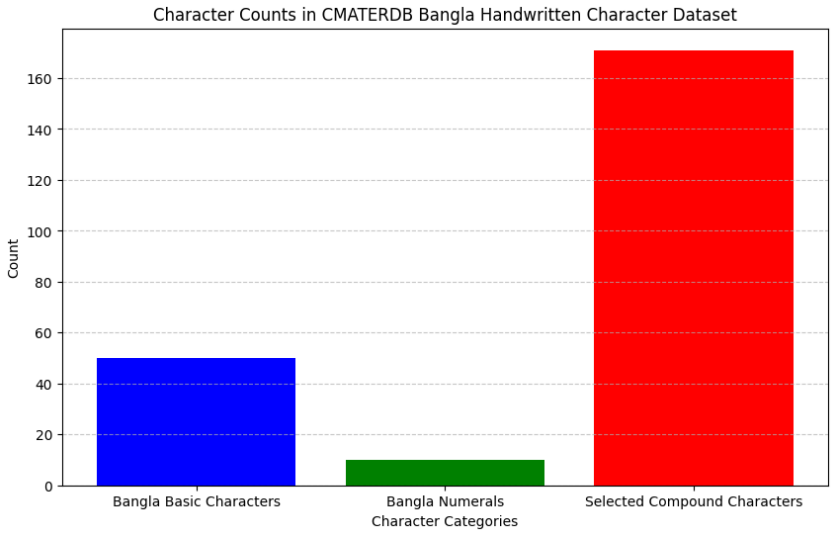


Figure 3.1: CMATERdb 3.1.3.3 Dataset Barchart

3.1.2 BanglaLekha-Isolated Dataset

BanglaLekha-Isolated is one of the cleanest Bengali handwritten character datasets available currently. This dataset was created by the Computer Science and Engineering Department of the University of Liberal Arts, Bangladesh, and the University of Asia Pacific, Bangladesh (Biswas, 2017)[22]. It was also funded by the ICT Division of the Ministry of ICT, Bangladesh. BanglaLekha-Isolated consists of 84 characters

of which 50 are basic characters, 10 are numeral characters, and 24 are compound characters. This dataset contains more than 1900 images for each character and contains a variety of handwriting in it. It also stored the age, gender, district, etc. of people while collecting the data so that research based on these characteristics can be done.

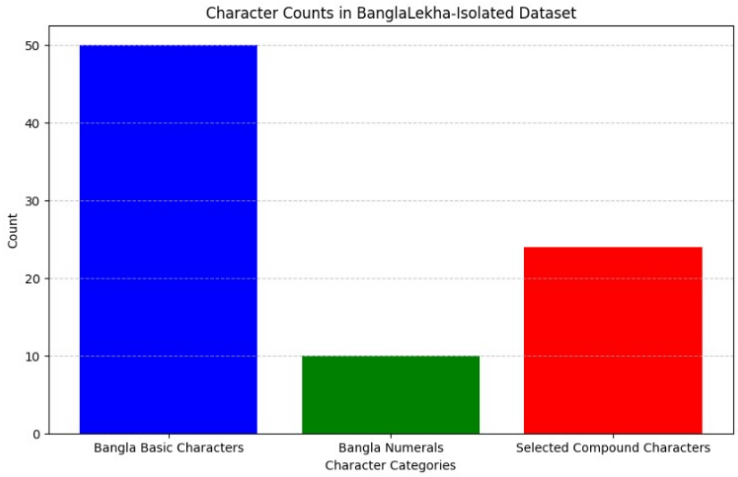


Figure 3.2: BanglaLekha-Isolated Dataset Barchart

3.1.3 MatriVasha: Compound Character Dataset

The "MatriVasha: compound character dataset" is an extensive compilation of Bangla handwritten character images, which includes both simple and compound characters [23]. This Bangla handwritten character dataset's usefulness is not limited to only handwritten character recognition. There are several other applications as well, such as writer gender prediction, age prediction and location identification. There are a total of 122 different characters in the dataset. Among them, 50 characters are basic characters of Bangla language, 10 characters are Bangla numeral characters, 52 compound characters and 10 are Bangla Sharachinno. Each unique character has a unique class for them, with more than 1200 images in each class and the collection is recorded in jpg format. Moreover, each image in the dataset is stored with a unique name which includes comprehensive details of the writer who wrote the character such as gender, age, location and occupation. This dataset is an invaluable resource in the field of Bangla handwriting recognition.

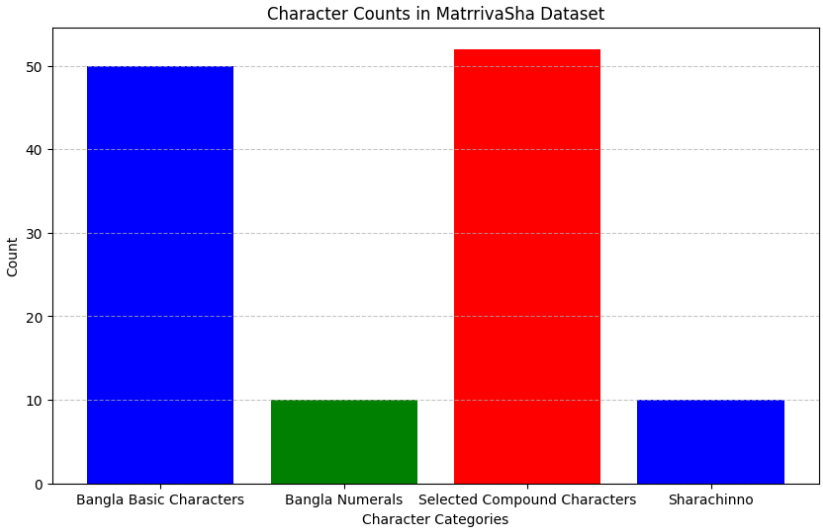


Figure 3.3: MatriVasha: Compound Character Dataset Barchart

3.2 Limitations of Existing Datasets

3.2.1 CMATERdb 3.1.3.3 Dataset Limitations

The CMATERdb 3.1.3.3 dataset is very popular and large. Though the CMATERdb 3.1.3.3 dataset provides a diverse collection of compound characters that are frequently utilized in the Bengali language, there are some flaws in the dataset which was creating an obstacle in achieving efficient accuracy. Many of the images in the dataset are problematic which made our job very difficult. Some images were very vague and some letters were unclear which made it hard for the models to learn. Again, some parts of the character were missing from the images which made the accuracy lower. Furthermore, as the dimensions of the images are very low, it is simply too blurry and unsuitable for the dataset to be used for the YOLO object detection model.

3.2.2 BanglaLekha-Isolated Dataset Limitations

Though Bangla-Lekha-Isolated contains a variety of handwriting for each character, it contains only 24 compound characters. Whereas, the number of compound characters used in the Bengali Language is far greater than that. This makes the ML/DL models trained on this dataset vulnerable to new compound characters that are not available in this dataset.
Another problem is, that some of the handwritten characters are incorrect and some of the characters are unclear in this dataset which may cause the models' lower accuracy. Some of the examples are shown in Table 3.1.

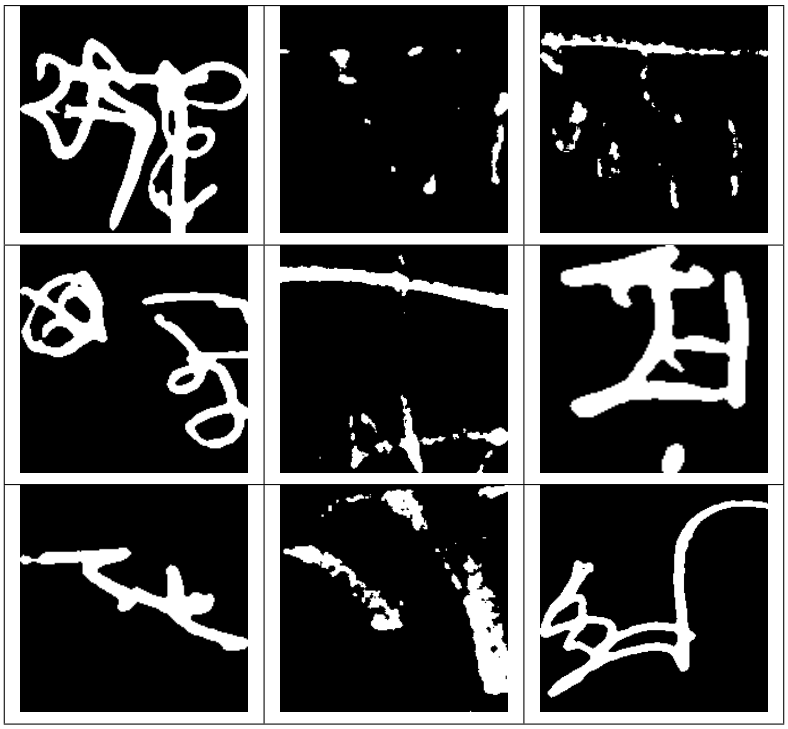


Table 3.1: Incorrect images inside BanglaLekha-Isolated Dataset

The above characters have no existence in the Bengali language and they were the result of incorrect handwriting or pre-processing and these characters may cause errors in the prediction models.

3.2.3 MatriVasha: Compound Character Dataset Limitations

The "MatriVasha: compound character dataset" has several applications in the field of nlp but the number of compound characters is very low. The dataset only has 52 compound characters, which can be called its biggest disadvantage as the number of compound characters in Bangla language is more than 3 or 4 times of this number. As a result, the machine will be unable to detect a large number of compound characters. Again, even if it has a great number of images in each class, there are also a good number of images that are unrecognizable. There are also some images that are misplaced. Moreover, different images in this dataset have different dimensions, and they are all of very low resolution, which makes them unsuitable for annotation and YOLO object detection.

Table 3.2: Incorrect images inside MatriVasha: Compound Character Dataset

3.2.4 Solution

To address these issues, we decided to create a new dataset. By creating this new dataset we aimed to overcome the flaws of CMATERdb 3.1.3.3 dataset and train our model with more clear and precise sample images.

3.3 Proposed Dataset

3.3.1 BanglaBorno Dataset

BanglaBorno Handwritten Bangla Characters dataset is a new dataset created by the authors of this research paper. The motivation behind the construction of BanglaBorno is to overcome the flaws that CMATERdb 3.1.3.3 dataset contains. We achieved this goal by collecting clear handwritten samples. BanglaBorno contains 50 simple characters and 162 compound characters. This led us in achieving improved accuracy results through VGG-16 architecture and YOLO classification and object detection.

3.3.2 Database Creation and Label Assignment for BanglaBorno Dataset

As the amount of compound characters in Bangla are huge, creating the database of compound characters was a challenging task. Initially, we stored the 50 simple characters labeled from 1-50 and after that, we started storing the compound characters. After storing the most common compound characters used in Bangla, the amount of compound characters was not very large. Then we started analyzing the

compound characters in the existing dataset and stored the ones which are used in Bangla language. After analyzing the existing datasets, we were left with 162 compound characters and 50 simple characters. Label 51-226 were the compound characters. A part of our dataset is shown below for better understanding.

Label	Character
90	ষ্ফ
91	জ্ঝ
92	দ্দ
93	ভ্র
94	দ্গ
95	ক্ষ্ণ
96	ল্ম
97	ন্স
98	শ্ম
99	স্ত

Table 3.3: Part of the database of BanglaBorno dataset

3.3.3 Data Collection for BanglaBorno

In the process of assembling our dataset, we initiated by creating a structured template using Adobe Illustrator software. This template consisted of multiple slots which were demarcated by black borders, arranged in multiple rows and columns. We utilized these borders to clearly define boundaries for each character so that the volunteers can easily understand how much space they have to write a character and also based on these boundaries we will split the a4 paper and have many images of a character. Subsequently, we saved this Illustrator file as a PDF of A4-sized paper. The next step involved printing multiple copies of this PDF template, which we distributed to students at Brac University. We requested the students who were our coursemates in different courses to provide handwriting samples by writing a specific character within each designated slot on the paper. To ensure clarity and organization, we labeled each sheet according to our database to indicate that only one particular character should be written on that page. This process was replicated for various characters, thereby yielding a diverse range of handwriting styles.
Also the assistance by our classmates, who helped by writing the assigned characters on multiple papers, allowed us to collect a wide array of handwriting samples for each character. Consequently, this procedure enabled us to gather a substantial amount of data for our dataset.

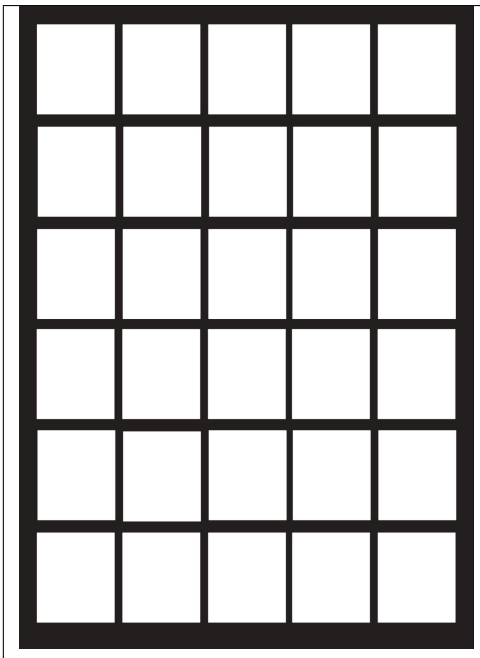
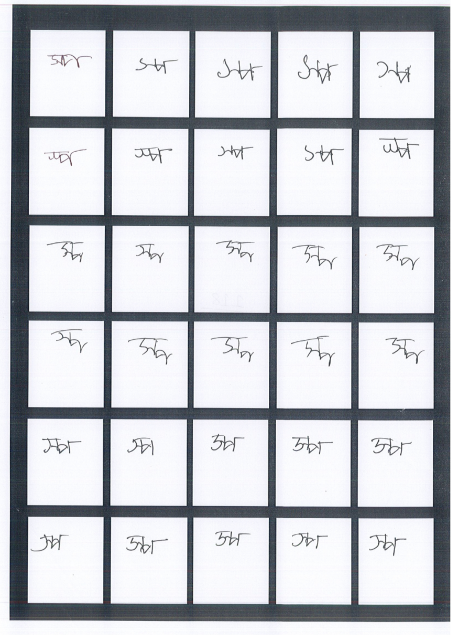
	
Blank	Handwritings

Table 3.4: Structured template for writing character

The dataset has the potential to give better performance for researchers and developers seeking to train and evaluate models designed to recognize handwritten Bengali compound characters who faced similar problems like us while using CMATERdb 3.1.3.3 dataset. The dataset's image quality, equal distribution and annotated labels make it a valuable resource for enhancing the current level of knowledge in this field. The aforementioned enables the creation of precise and resilient mechanisms for identifying compound characters in practical Bengali handwriting situations.

3.3.4 BanglaBorno Preprocessing

Once we had gathered the data, our next step was data processing. We secured permission from our supervisor, Mr. MIH, to utilize the scanner located on the 8th floor of Brac University's UB8 building. Given the large number of samples we had collected, we conducted scanning in multiple sessions and generated PDF files.

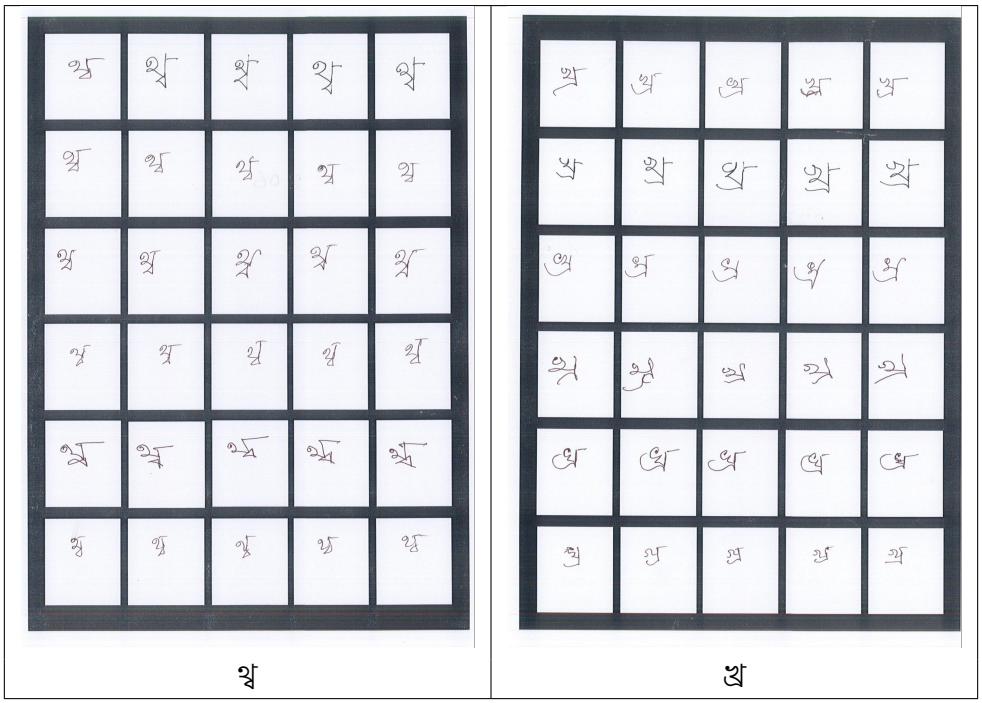


Table 3.5: Samples of the Collected Data

These PDF files were then transferred to our personal computers. To facilitate further processing, we initially converted the PDFs into JPG format. Subsequently, we designed a method to segment each page into individual images, taking into account the multiple rows and columns of characters present on each page.
After analyzing the images, we found that certain letters are written with very light and thin strokes. In order to overcome these challenges, we were going to need to preprocess our dataset and come up with a workaround. Because some of the photographs lacked clarity, we needed to make the pictures or the pixels darker. We came up with a method of binarization that is not only efficient but also very speedy.
To ensure uniformity and enhance the clarity of the dataset, we transformed black pixels to white and vice versa, effectively segregating the various handwriting styles for each character into separate images. For that, we used a thresholding method to convert the white pixels to black and the black pixels to white. First, the pixel values greater than 200 are converted into 255 and lower than 200 are all converted to 0. In this method, if a pixel is already quite black, it will become darker, and if it is already predominantly white, it will become completely white. This will allow the writing and pen strikes to be seen more clearly. When we used this method, we obtained outcomes that met our expectations.
After that, we inverted the process and made the white pixels black and the black pixels white.

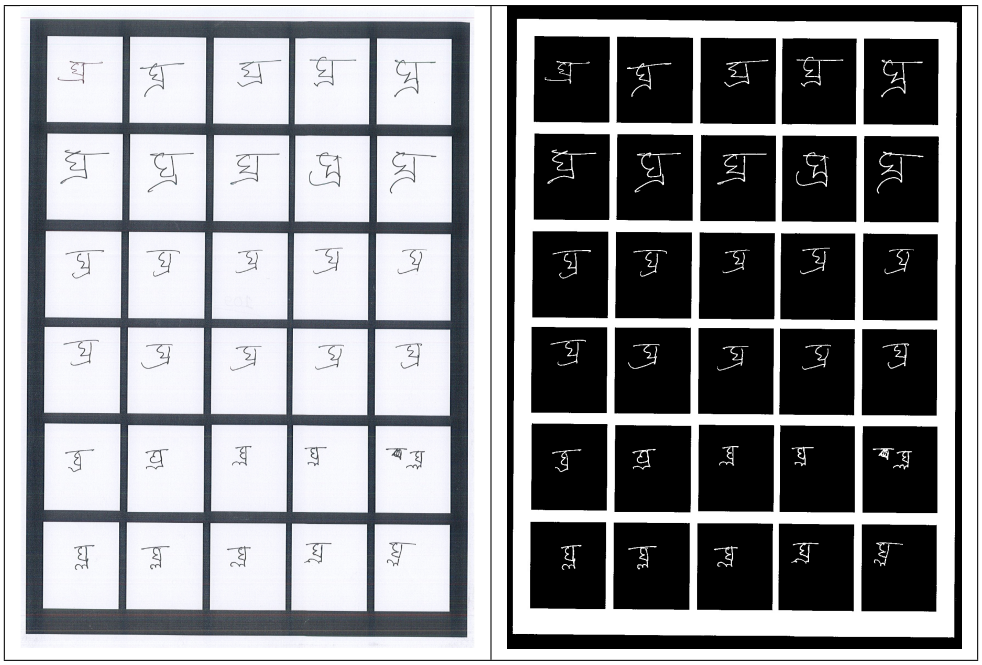


Table 3.6: Before and After Thresholding

After that, we imported the necessary libraries for processing the image. Then we converted the image to a numpy array, cropped the image to remove white borders and returned the cropped box image which is stored in the specified output file.

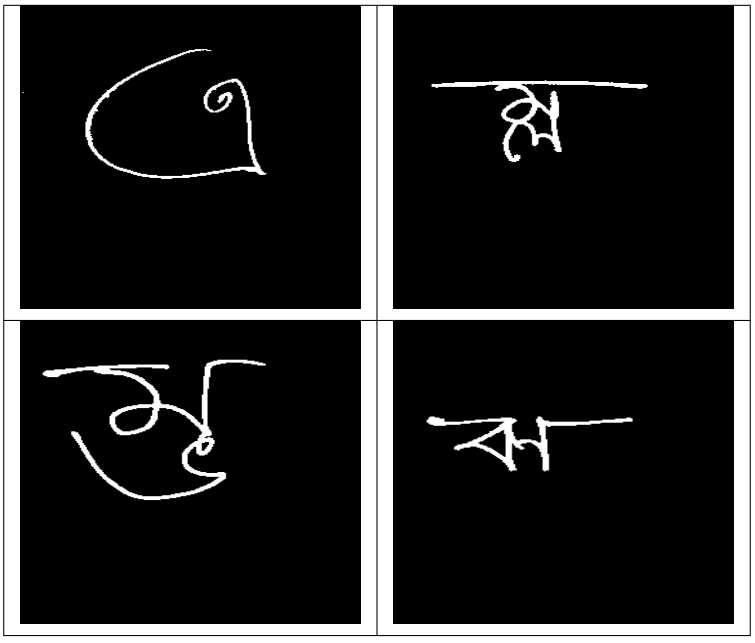


Table 3.7: Cropped character images from collected data

The images shown in Table 3.6 were then organized into folders based on the character they represented. In total, we established 212 classes of handwritten characters, each corresponding to a unique character, thereby thoroughly categorizing our dataset.
Furthermore, corrupted images and incorrect characters needed to be removed. Therefore, we were required to manually check each of the 212 classes to remove

those images and characters and ensure there are no unnecessary images. This is to make sure that the machine does not learn with any unnecessary characters which could reduce the performance. After that, we moved all of the classes into a folder named Train and created another folder named Test folder containing the same classes. Then, we transferred 10% of images from each class within the Train folder to the corresponding class within the Test folder. Thus, we created a Train and a Test folder in our dataset.

3.3.5 Dataset Informations

Train	Test
10,095	3,713

Table 3.8: Train and Test image count of BanglaBorno Dataset

The images of this dataset were obtained at a resolution of 96 dots per inch. Owing to the use of a document scanner the images of our dataset have very precise characters with reduced distortion and image noise. The dataset has a total 13,808 images, 212 annotated folders containing.

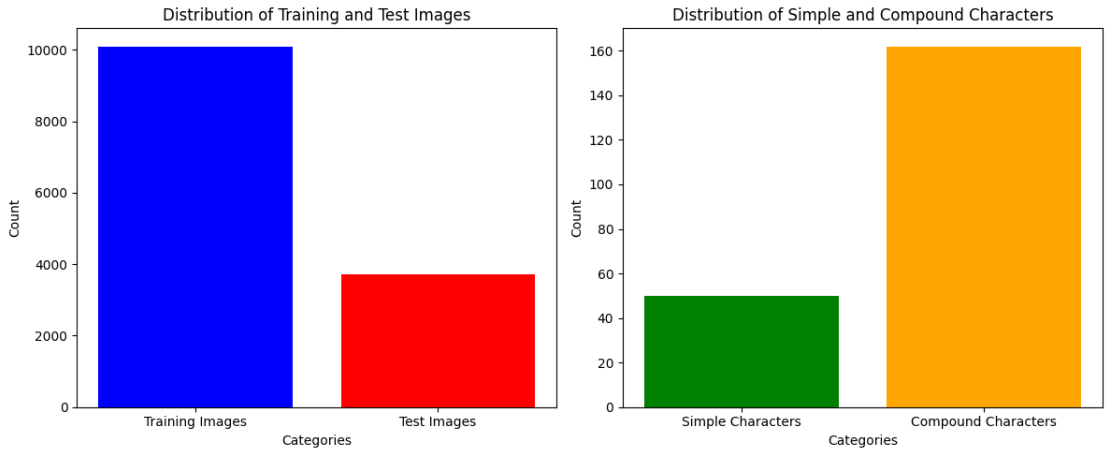


Figure 3.4: BanglaBorno Dataset Informations

The dataset has the potential to give better performance for researchers and developers seeking to train and evaluate models designed to recognize handwritten Bengali compound characters who faced similar problems like us while using CMATERdb 3.1.3.3 dataset. The dataset's image quality, equal distribution and annotated labels make it a valuable resource for enhancing the current level of knowledge in this field. The aforementioned enables the creation of precise and resilient mechanisms for identifying compound characters in practical Bengali handwriting situations.
When we started experimenting with the dataset, we discovered a few characteristics of our dataset. In Bangla, one compound character can be written in multiple handwritings. Our dataset contains different kinds of handwriting styles of different people which will make the OCR perform better.

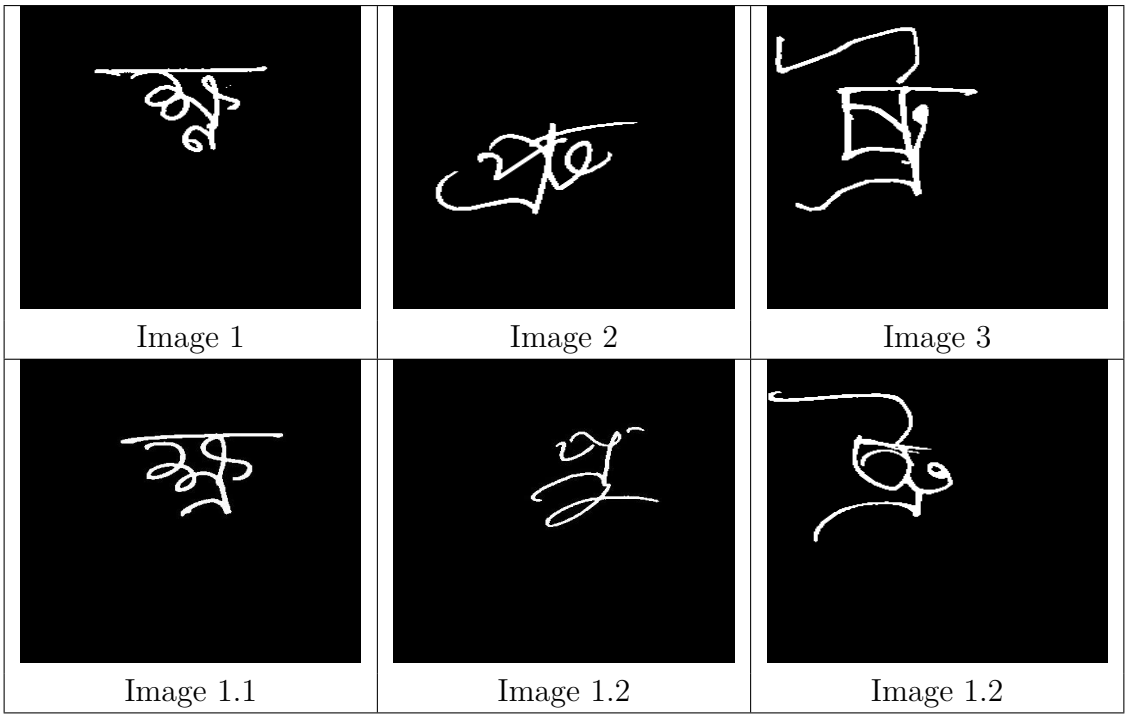


Image 1	Image 2	Image 3
Image 1.1	Image 1.2	Image 1.2

Table 3.9: Different writing style of same compound character in BanglaBorno Dataset

3.3.6 Augmentation

Because of time constraints, the amount of images in the dataset is not very large. But we needed to counterfeit overfitting or underfitting before using our model. In order to do that, image augmentation was performed with the rotation range of 5, width_shift range of 0.08, height_shift_range of 0.08, shear_range 0f 0.2 and zoom_range of 0.2. Each folder had a maximum of 400 images in the training set.

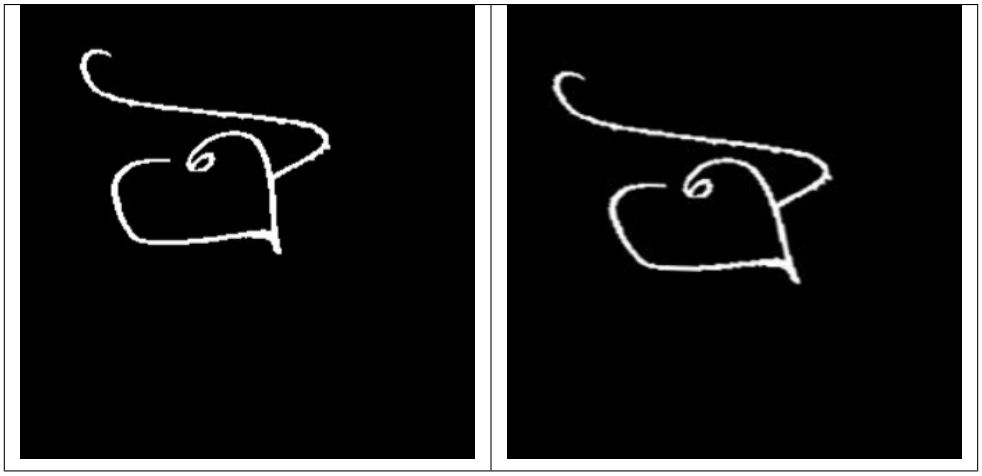


Table 3.10: Before and After Augmentation

After doing the augmentation, the number of images in each class had become 400 and it was a balanced number of images in order to train the models. These preprocessing processes reduced biases, improved model performance, and captured a wide range of real-world writing patterns.

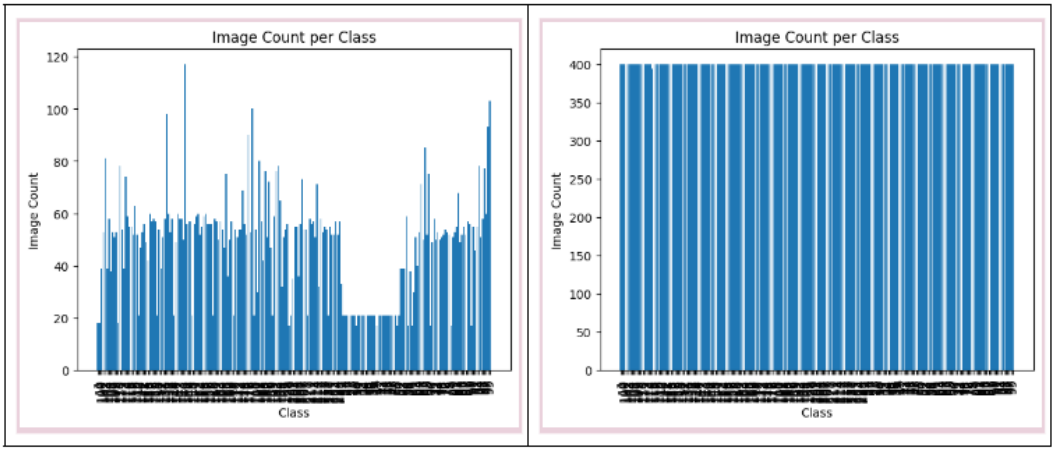


Figure 3.5: Images count per Class before and after Augmentation

3.3.7 Comparison with BanglaBorno

	Cmaterdb	BanglaLekha-Isolated	MatriVasha	BanglaBorno
Compound characters	171	24	52	162

Table 3.11: Number of Compound characters in the Datasets

Chapter 4

Model Employment & Preprocessing

4.1 Pre-processing for VGG-16 and YOLOv8 Classification

4.1.1 Image Segmentation

Most of the compound bangla characters are built in such a way that they can be split either horizontally or vertically and separate the simple characters. Thus, our first approach was to segment the compound characters into the top, right, left, and bottom. So to do this we created a bounding box, made the character's color white, the background black color, and divided the character into a total of 4 parts. We kept the 4 parts (top, right, left, and bottom) in 4 separate folders labeled top, right, left, and bottom. Now our next step was to train these data via VGG-16 and YOLO classification models.

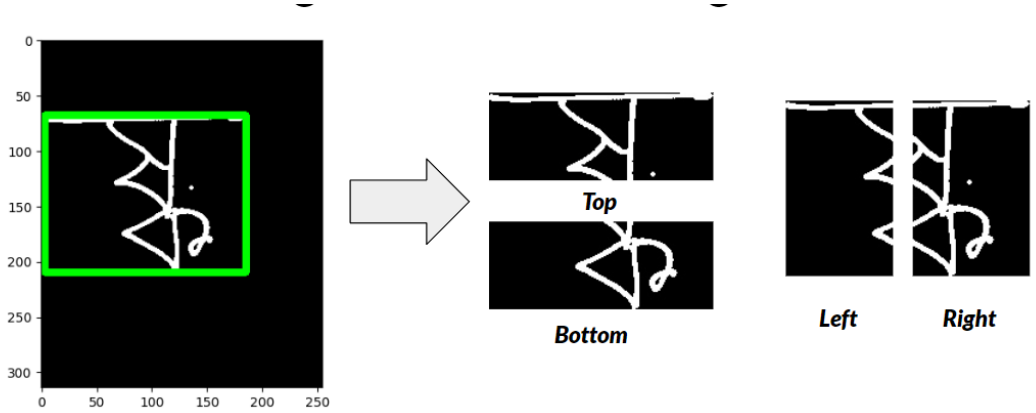


Figure 4.1: Drawing bounding box and Segmentation

4.1.2 Database creation

To categorize compound characters we labeled each row corresponding to specific labels. For instance, for Cmaterdb folders 1 to 171 represent compound characters, and thus we created a column with the folder names of each character 1-171 representing the compound characters and the next 60 column values representing simple

characters and digits from 0 to 1. Each row contained the information for a specific label. We individually went through the content in the folders corresponding to the top, right, left, and bottom positions, analyzed the characters for these respective folders, and identified the most frequent characters for each of them.
For our dataset, **BanglaBorno**, we followed the same method but the digits and numbers of characters were different this time. Labels 1-50 represented the simple characters, 51-60 represented the characters that could not be split and 61-226 represented the compound characters. Like before, Top, Bottom, Left, Right, and Total columns represented their respective parts of characters in their respective parts of the image.

Serial	Top	Bottom	Left	Right	Total	Character
96	ত	ন	ত+ন	ত+ন	ত+ন	ত্ন
97	ষ	ক্র	ষ+ক্র	ষ+ক্র	ষ+ক্র	ষ্ক্র
98	ঘ	র-ফলা	ঘ+র-ফলা	ঘ+র-ফলা	ঘ+র-ফলা	ঘ্র
99	দ	ভ	দ+ভ	দ+ভ	দ+ভ	দ্ভ
100	শ	ল	শ+ল	শ+ল	শ+ল	শ্ল
101	ব	ধ	ব+ধ	ব+ধ	ব+ধ	ব্ধ

Table 4.1: Database for Compound character information

Let's consider label 96 as an example representing the compound character "ত্ন". We first inspected the characters in the top folder and saw that the majority were "ত", thus under the Top column for label 96 we wrote "ত". Again, for the bottom folder, we saw that most of the characters were "ন", so we wrote "ন" for label 96 under the column bottom. However, when checking the right and left folders we found that both the characters were split in half and thus wrote "ত + ন" to denote that the compound character is composed of these two simple characters. We performed the same process for all the labels and filled up the Google Sheet for all the labeled rows. So the sheet contains the complete information, accurately reflecting the characters' distribution across the different positions. We used this systematic approach to check the compound characters and how they are split within the images across the top, right, left, and bottom folders and match with the result that we got after training our model.

4.2 VGG-16 Model Employment

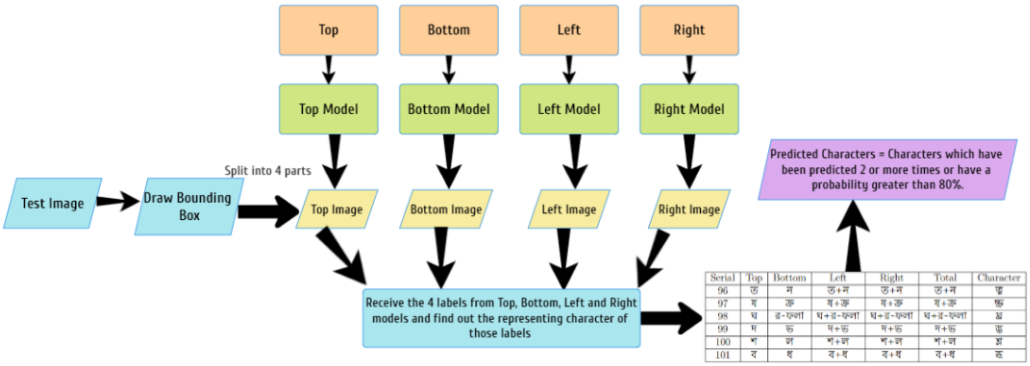


Figure 4.2: VGG-16 Classification implementation Flowchart

For this research to perform character localization and recognition the VGG16 model is used with the purpose of focusing on character identification in 4 different folders named top, right, left, and bottom respectively. For this character identification, we trained four separate VGG-16 models for the four folders, each trained to predict characters in their respective positions. Each character has 500 images in the 4 folders respectively for training. After training the models we used them to detect the characters accurately. So after loading the model separately for the four different folders the models gave predictions for each of the folders for the test image like what is the top part, what is the bottom part, and so on in the test image. Then after getting the four different predictions from the four models for top, bottom, left, and right respectively, we checked in the sheet we made before for the characters. For example, if the top model gives prediction 11 then we checked that under the top column for row 11 it has the character "গ" and if the bottom model predicts the label 4 then in the sheet it has "র-ফলা" for row 4 column bottom. The left and right model predicts label 11 and in the Google sheet we saw that it is "গ+র-ফলা". As this character cannot be separated horizontally but vertically, the left and right folder contains both characters. When the character cannot be separated horizontally but it's separable vertically then the right and left models may give the wrong output and when it is not separable vertically but horizontally then the top and bottom models may give the wrong answer. Even though the bottom model did not predict label 11, it predicted the label which is the same character. Now from analyzing the given predictions, we see that the appearance of 11 is more than 2 times so the character must be a compound character in label 11 and also by cross checking the predictions with the sheet, we can say that the given test image is "গ+র-ফলা"/"গ্র" which is the compound character of label 11 and it is the accurate answer. By applying this process using VGG16 we get 60% accuracy.
The approach of splitting the characters vertically and horizontally does not work for simple characters. As splitting the simple character does not give any meaningful character, the performance of our model was reduced and the accuracy result was not satisfactory. So to avoid facing this issue we needed a way to only split the compound characters and not the simple characters as it would make the images meaningless. Thus, we adopted a new strategy for training the dataset. In the top, bottom, left and right folders, compound characters were split like the below image

but the simple characters were kept in their original shape in all these 4 folders. Another folder was created to train the simple characters only. As a result, we were left with 5 folders, Top, Bottom, Left, Right, and simple characters and we needed to train 5 different models for the total prediction.

Figure 4.3: Top, Bottom, Left and Right images of a Compound character

For prediction, we had initially employed the VGG16 model for character localization however it gave an unsatisfactory result of 60% accuracy. That's why we performed the same process using the YOLO (You Only Look Once) v8 classification hoping for a better accuracy.

4.3 YOLO CLassification

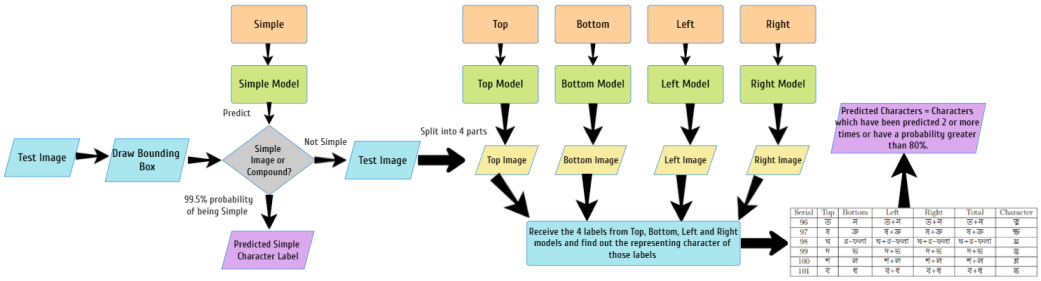


Figure 4.4: YOLO Classification implementation Flowchart

YOLOv8 is a more recent and faster image classification model that has brought a new revolution. After the unsuccessful approach in the VGG-16 model, we changed our strategy slightly. We initially trained the YOLOv8 model with characters that should not be split into four parts. For example, simple characters should not be split into four parts, or else they cannot be identified after splitting. There are also some compound characters that should not be split. Some of these characters are shown below.

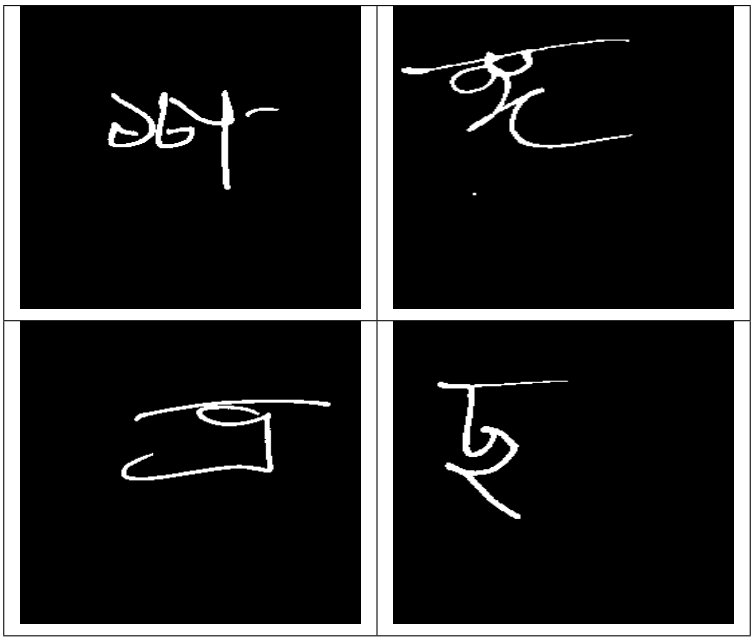


Table 4.2: Simple characters and Compound characters that should not be splitted

So, a YOLOv8 model was trained on the images which cannot be segmented. This model is referred to as 'Simple Model' here. Then, another four YOLOv8 models were trained with the Top, Bottom, Left and Right part of the images sequentially. We begin the prediction of the test images by examining whether a character can be classified as "simple" or not by using the model trained with simple characters only. If this model is at least 99.5% confident about it's prediction, then we classify the character as simple and thus we proceed to predict the image using the simple model. We retrieve the predicted label and cross-reference it with our database to determine which character corresponds to that label and compare this predicted character with the actual class label character.
In cases where the initial check does not yield a 99.5% probability then we assume that the character is not simple and we proceed with more detailed recognition considering it as a compound character. We split the test image into four parts: top, bottom, left, and right. We then employ separate four models for each four parts to make predictions. Then we analyzed the predictions and identified which labels were predicted two or more times across the four parts. Subsequently, we reference our database to identify which characters are in these prediction labels and append them to a list. With the list of predicted characters in hand, we compare them to the actual label characters to assess the accuracy of our predictions and find out if any of the predicted characters match the actual characters present in the image.

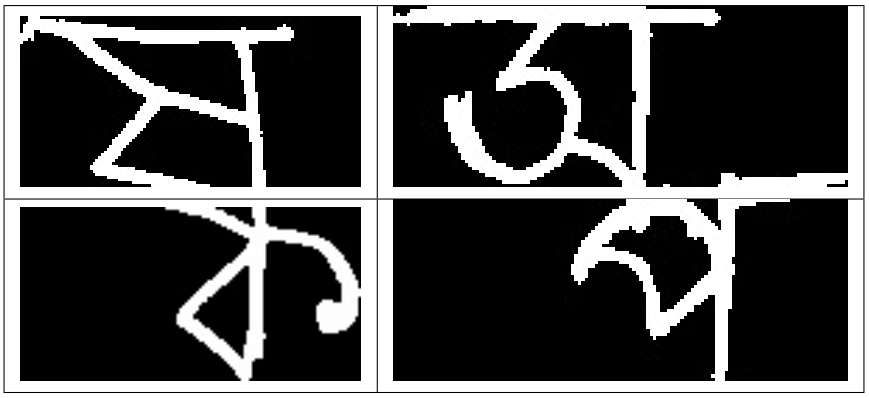


Table 4.3: Top Bottom Split

To demonstrate it more, a picture of the character number 101 after splitting into top and bottom has been shown here. Now, suppose a test image has been split into the top, bottom, left, and right parts of the image and the top model gives the predicted label as 101. Then we fetch from the database the representing character at the top of 101 and understand that there is a 'ষ' at the top of the image. After that, the bottom model gives the prediction label as 118. Using the same method we find that the bottom of 118 is 'প'. So, we can conclude that the whole character is 'ষ + প = ষ্প'.
Furthermore, in some cases where no characters are predicted more than once, we consider the individual probabilities of top, bottom, left, and right parts and which characters have probabilities exceeding 50%. Then we stored the probabilities of the characters that exceeded 50% probability and included these characters in the main prediction if they either occur twice or more or if their probability exceeds 80%. Following this, we identify the characters according to the prediction and again compare them to the actual label characters.
This multi-stage approach allows us to optimize character recognition accuracy. We first check for simple characters and, if applicable, use a dedicated model so that it does not reduce the accuracy. For more compound characters, we break down the image into four parts and incorporate many steps to get the final prediction. Additionally, we consider individual character probabilities when necessary to improve prediction accuracy.We attained a maximum accuracy of 81.8% with this multi-stage approach.
Using YOLO Classification Model, we got better results but it still was not up to the mark. In order to find out the reason behind this lower accuracy we conducted an analysis and discovered that some of the images were not perfectly split. As we drew a bounding box and split it in two, some handwriting was not able to split properly. So, we tried our next method, YOLOv8 Object Detection Model for a better result.

4.4 Pre-processing for YOLOv8 Object Detection

4.4.1 Structuring Filename

In order to keep track of all the main classes and subclasses of each character we needed an efficient way for both training and evaluation testing. So we followed the

class names sheet and renamed all images with classnames first.

4.4.2 Label Creation and Annotation

Now we needed a YAML configuration file for training with unique class names and labeled images with those classes. So we carefully determined all the unique simple character classes and gave them a unique digit number (For Example 39 to ল) and completed the annotation process.

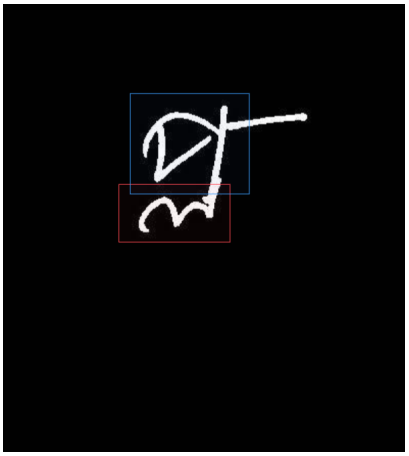


Figure 4.5: Annotated Image by class

4.5 YOLOv8 Object Detection Model Employment

In our research, we have used YOLOv8 Object Detection to detect individual simple characters from the compound characters. We have used our own dataset **'BanglaBorno'** for training and testing. Firstly, we randomly chose 5 images from each of the 162 compound character classes and combined them in a folder for annotation. After annotating each image using CVAT manually we had the training images and corresponding label files containing the coordinates of the boxes.

0	0.554744	0.443414	0.583114	0.282298

Table 4.4: Training image label

Then we prepared a YAML Configuration file with train data location and number of classes and their names. Now that we have everything we need for training, we figured out that if we use the simple character classes in our training that would give us much better results. So, we added the rest of the 50 classes and prepared them for the final training.

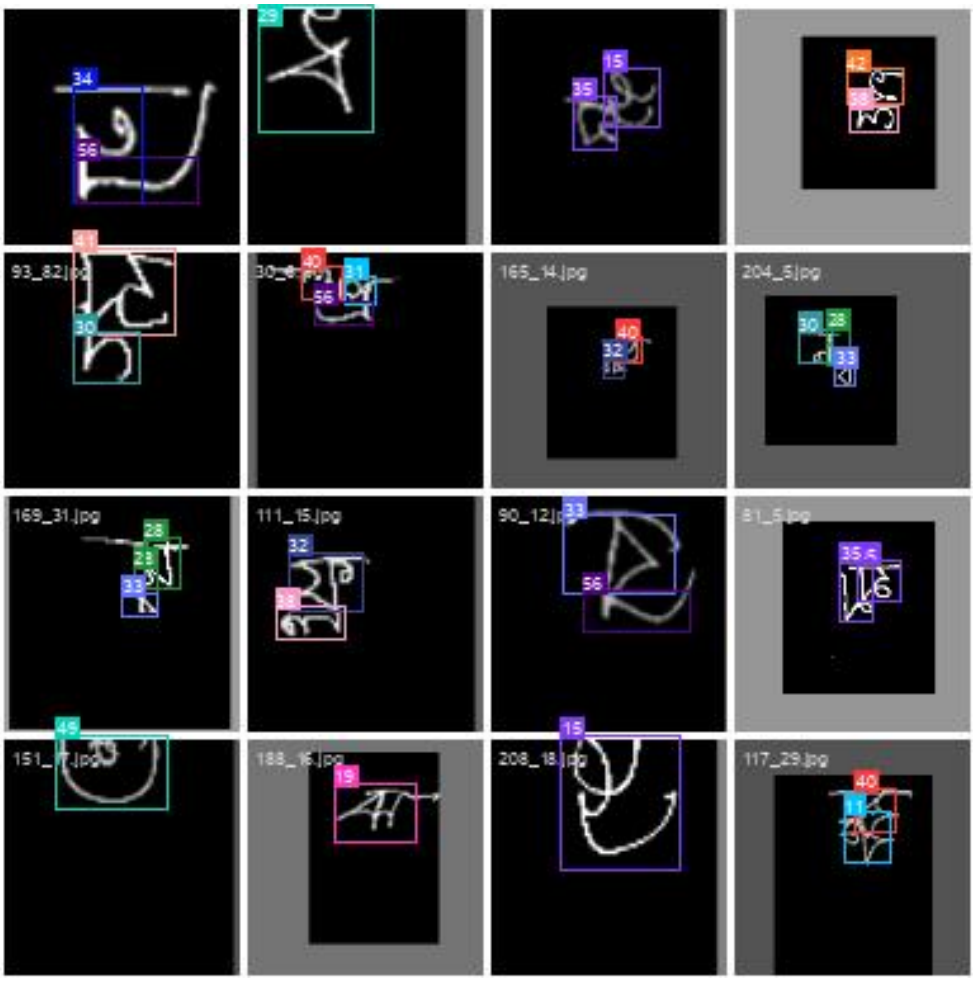


Figure 4.6: YOLO Training Batch

Once we are done with the training process we need a way to evaluate. In order to fulfill our requirements and control over the whole testing process with accuracy by class we came up with our own testing method. We simplified the idea into parts. Firstly, we prepared a test set from the rest of the images that we hadn't used for training in a format that has been renamed classnunber_serial.png format. The renaming part is simple but crucial for the whole process.
Upon running the test files we found the images with predicted label names and confidence score

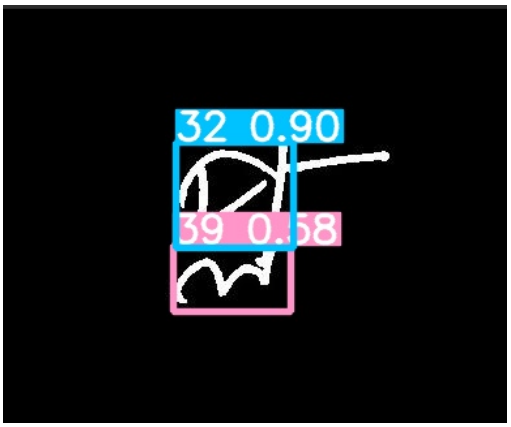


Figure 4.7: Predicted Classes with confidence score

Next, we modified YOLO to save the predicted results of each test image in a text file with predicted classes.

39	0.450882	0.452851	0.2267	0.111842
32	0.455919	0.3125	0.2267	0.177632

Table 4.5: Predicted Results in TXT File

So, we had a result text file with the same name as the test image for all the test images to move to the next progress.
Now, we needed to compare the test result with the actual results for that corresponding image for evaluation. To achieve this, we needed to make another database that should contain all the sub-simple character classes of a compound character class formatted so that we can use the data to compare between the subclasses.

```
{
  "100": ["35", "26"]
}
```

Table 4.6: JSON Database of Main and Subclasses

Finally we are ready to evaluate. We run through each results file and read it's data to find the detected subclasses, then, we read the compound class name from the text file as it has been renamed accordingly. Once we have the main compound class name and predicted subclasses name, we fetch the actual subclasses name using the main class name from the database and compare predicted subclasses and actual subclasses.

main_compound	=	'100'
actual_subclasses	=	[35, 26]
predicted_subclasses	=	[35, 26]

Table 4.7: Evaluation Process

Finally, after evaluation, we achieved 82.44% accuracy. The accuracy percentage is satisfactory considering the limited number of images we used for training. Even though we used 5 samples from each class, more data from each class was trained because there were the same simple characters in multiple compound characters. The simple character ক was in both ক্ল (ক + ল) and ক্ট (ক + ট) which helped in the training process.

Chapter 5

Result & Analysis

5.1 VGG-16

After using VGG-16, we could not find our desired result but we understood different limitations in our approach and worked for the improvement of our approach using other models.

1. Challenges in differentiating simple and Compound characters: Initially, all of the train images were split into 4 parts and it also split the simple characters into 4 parts. It created meaningless information in the case of simple characters. This led us to find a solution for differentiating simple and compound characters.

2. Accuracy: By using VGG-16, we achieved an accuracy of 60%. One of the reasons behind the lower accuracy was the split of simple characters. Another factor that worked behind this was the lower quality of the image dataset after pre-processing.

The validation accuracy graphs of the Top, Bottom, Left, and Right are shown below.

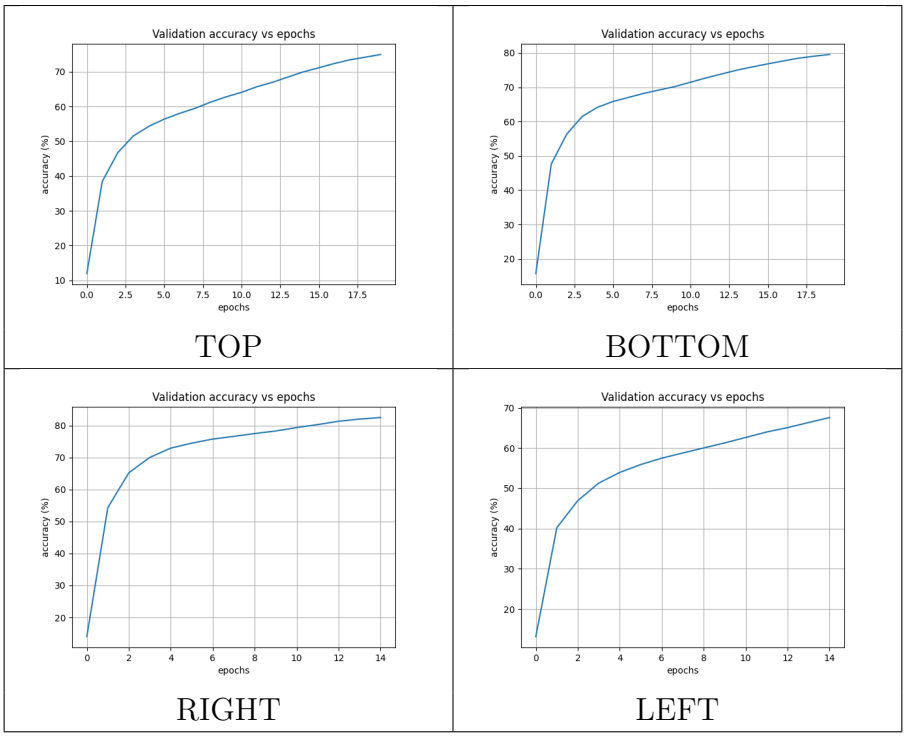
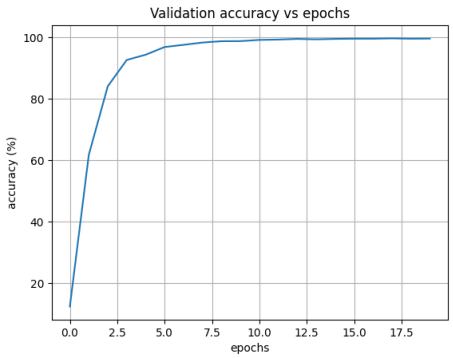


Table 5.1: Validation Accuracy for the four VGG-16 models

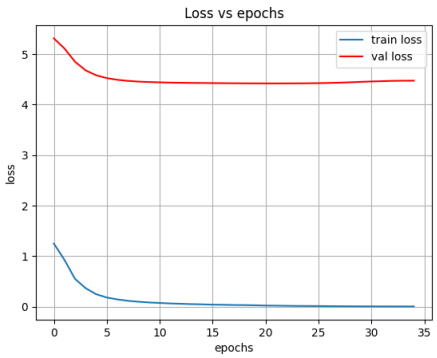
From the four validation accuracy carves of VGG-16, we can see that the VGG-16 models and our methodology could not give the expected results properly.
3. Training Strategy: The strategy of segmenting the characters into 4 parts Top, Bottom, Left and Right could solve the problem of detecting the underlying simple characters most of the time.

5.2 Yolov8 Classification Model

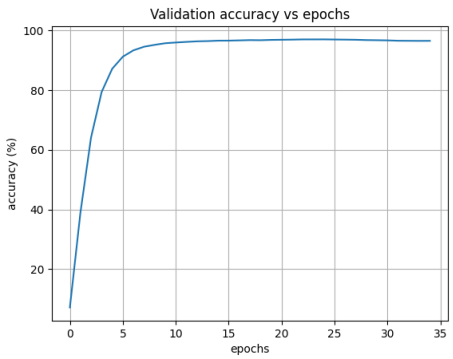
After using the YOLOv8 classification model, we managed to overcome the problems we faced during VGG-16. But we also observed some new problems that prevented our model from achieving greater accuracy.
1. Successfully differentiated Simple and Compound characters: While using the YOLOv8 classification model, we trained the simple characters without segmenting them and trained the compound characters using the same segmentation technique we used in the VGG-16 model. By using the simple model to determine if the test image is a simple character or not, we managed to prevent any simple character from getting segmented. Thus, the meaningless images that were generated during VGG-16 were not there anymore and it improved the overall performance.
2. Improvement of Accuracy: After using the YOLOv8 classification model and preventing the simple characters from being split, we managed to get an accuracy of 81.8%. Another factor which worked behind the improvement was our multi-step prediction method.
If we compare the validation curves of the YOLOv8 classification models, we can notice that it works much better than our previous VGG-16 model.



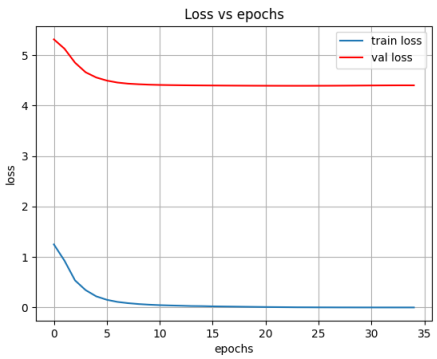
Simple Character Train Accuracy



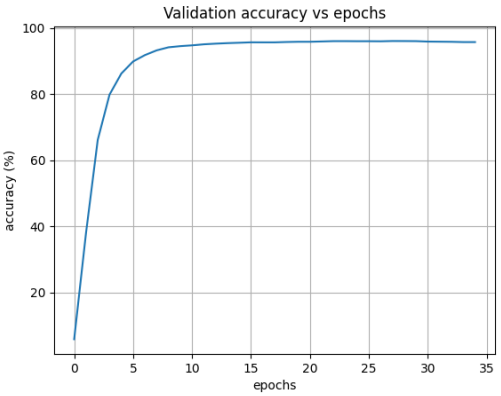
Simple Character Train Loss



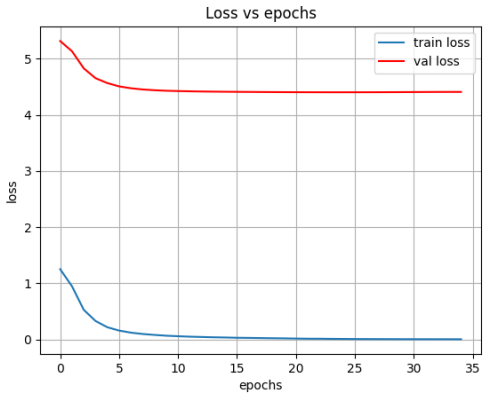
Top Character Train Accuracy



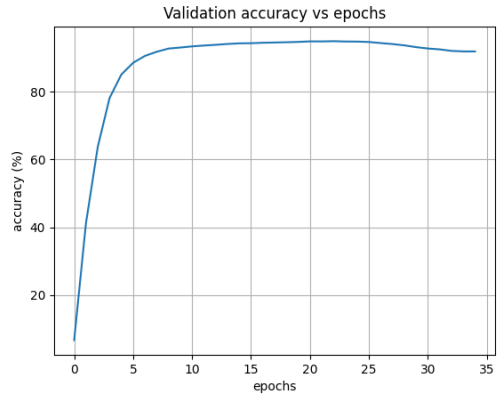
Top Character Train Loss



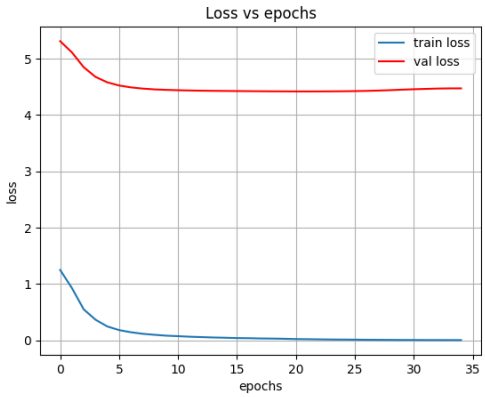
Bottom Character Train Accuracy



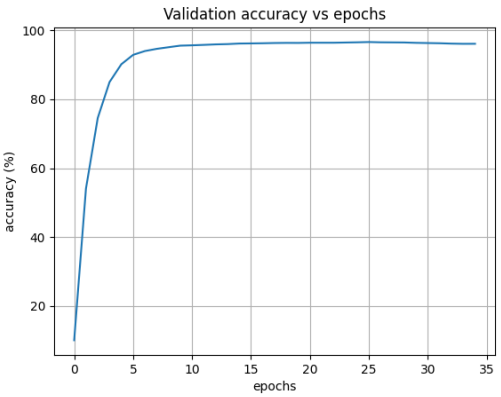
Bottom Character Train Loss



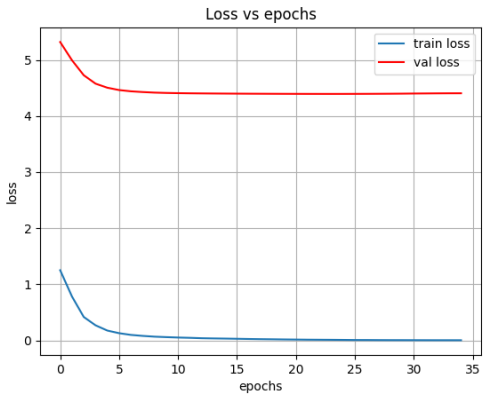
Left Character Train Accuracy



Left Character Train Loss



Right Character Train Accuracy



Right Character Train Loss

Table 5.2: Validation Carves for the YOLOv8 classification model approach

To visualize the accuracy for all the class labels, a graph of Class Labels vs Accuracy is given below.

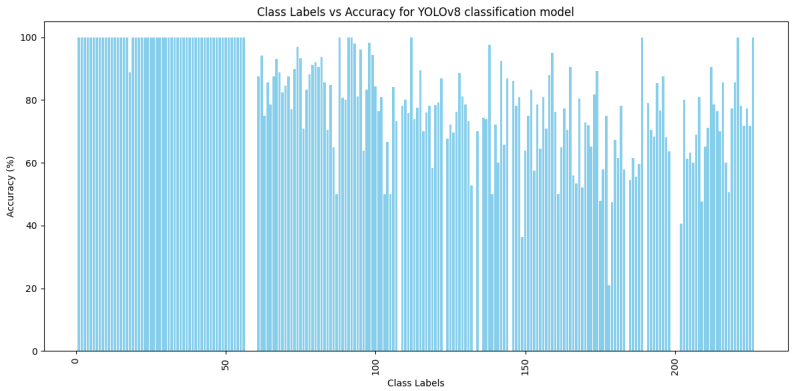


Figure 5.1: YOLOv8 Classification Accuracy visualization

3. Limitations: Even though we improved the accuracy of detection of underlying separate characters in the test images, the rate was not good enough. The main reason behind this was the uneven split of the characters like below. Also, the images that could not be split into top/bottom were still being split into top/bottom.

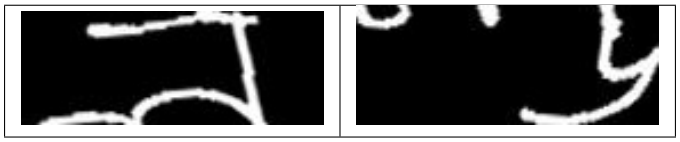


Table 5.3: Uneven Split of a character

To overcome the limitations of our training process, we approached a different version of YOLO which would solve the uneven split problem. We tried the YOLOv8 object detection model for predicting the underlying simple characters inside the compound characters.

5.3 Yolov8 Object Detection Model

With the YOLOv8 Object Detection Model, we achieved our goal of segmentation. Though we achieved higher accuracy and segmentation, we were limited by time and training resources.

1. Successful segmentation of Compound Characters: We have been able to successfully determine the simple characters that form the compound character from the image of the compound characters.

2. Improvement of Accuracy: With the YOLOv8 Object Detection model we achieved the highest accuracy of 82.44%. By labeling simple characters alongside compound characters in training, the model improved overall accuracy.

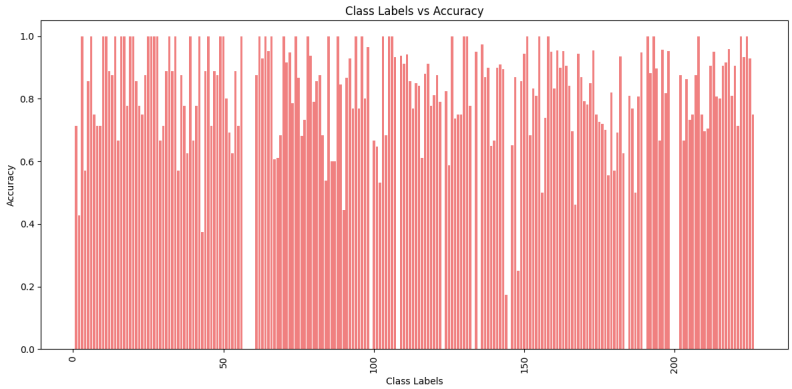


Figure 5.2: YOLOv8 Object detection Accuracy visualization

3. Limitations: Even though we could increase the accuracy and perform segmentation. However, due to time constraints, we could train only 5 images per class which gave us this satisfactory result. If we can increase the training images to at least 100 images per class then it is possible to achieve a much higher accuracy and confidence score.

5.4 Comparison of Models and Approaches

While evaluating the predictions, a prediction is considered correct if all the characters in the images can be predicted correctly. Otherwise, we consider the prediction as wrong.

Model Name	Accuracy	Position
VGG-16	60%	3
YOLOv8 Classification Model	81.8%	2
YOLOv8 Object Detection Model	82.44%	1

Table 5.4: Accuracy Comparison between models

After the three different approaches to our model, we conclude that the YOLOv8 Object Detection model performed better than the other approaches with 10 times less training data.

Chapter 6

Conclusion & Future Work

6.1 Conclusion

The existing Bangla OCR research does not apply segmentation for the detection of compound characters, but segmentation is capable of identifying the underlying simple character. It can also help partially detect unclear handwriting. In our research, we applied different methodologies for the segmentation of compound characters in order to detect the underlying simple characters. Furthermore, to overcome the limitations of the existing datasets, a new Bengali Handwritten Character dataset **BanglaBorno** has been developed. We performed multiple experiments to conduct this research and got the most satisfactory results from the YOLOv8 classification model and the YOLOv8 Object detection model. But our research still has the scope of betterment which can be done in the future.

6.2 Future Works

6.2.1 BanglaBorno Dataset

Our dataset, **BanglaBorno** contains images of 162 unique compound characters, which will be a help in Bangla OCR research. However, we hope to bring improvement in the dataset by adding more variety of handwriting. To do that we plan to collect the handwriting of the characters from people of different ages and locations. Thus adding more variety and numbers for each of the unique characters will bring a huge improvement to our dataset. Moreover, we plan to add more compound characters to the dataset.

6.2.2 YOLOv8 classification method

Before training the model, we processed our data by splitting every image of our dataset horizontally and vertically. In both splits, the bounding box of the characters was equally divided into 2 parts. However, the compound characters' placement, length, and handwriting styles were different for different images. As a result, in a lot of cases, the compound characters were not properly divided. Therefore, in the future, we plan to bring improvements in our splitting method by making sure the split is done according to the length of the compound characters and the handwriting style. This will also increase the performance of the model.

6.2.3 YOLOv8 Object detection

Lastly, one of the difficulties that we have experienced in the YOLOv8 object detection model is annotating the huge number of compound character images. The performance of the model depends on the amount of images that have been annotated. In order to enhance the performance of the model, more images of each class are needed to be annotated. Hence, our future plan includes the annotation of more compound character images as we expand the dataset.
Another improvement that could be done is, merging the three approaches together in order to get more accurate results. By using the probabilities and predictions of VGG-16, the YOLO classification model, and YOLO object detection model together, the final predictions can be a lot stronger.

Bibliography

[1] B. Purkaystha, T. Datta, and M. S. Islam, "Bengali handwritten character recognition using deep convolutional neural network," in *2017 20th International Conference of Computer and Information Technology (ICCIT)*, 2017, pp. 1–5. DOI: 10.1109/ICCITECHN.2017.8281853.

[2] B. Dessai and A. Patil, "A deep learning approach for optical character recognition of handwritten devanagari script," in *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, vol. 1, 2019, pp. 1160–1165. DOI: 10.1109/ICICICT46008.2019.8993342.

[3] J. Lane, *The 10 most spoken languages in the world*, Jun. 2021. [Online]. Available: https://www.babbel.com/en/magazine/the-10-most-spoken-languages-in-the-world.

[4] R. Pramanik and S. Bag, "Shape decomposition-based handwritten compound character recognition for bangla ocr," *Journal of Visual Communication and Image Representation*, vol. 50, pp. 123–134, 2018, ISSN: 1047-3203. DOI: https://doi.org/10.1016/j.jvcir.2017.11.016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1047320317302250.

[5] N. Das, K. Acharya, R. Sarkar, S. Basu, M. Kundu, and M. Nasipuri, "A benchmark image database of isolated bangla handwritten compound characters," en, *Int. J. Doc. Anal. Recognit.*, vol. 17, no. 4, pp. 413–431, Dec. 2014.

[6] S. Bag, G. Harit, and P. Bhowmick, "Recognition of bangla compound characters using structural decomposition," *Pattern Recognition*, vol. 47, no. 3, pp. 1187–1201, 2014, Handwriting Recognition and other PR Applications, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2013.08.026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320313003671.

[7] S. Roy, N. Das, M. Kundu, and M. Nasipuri, "Handwritten isolated bangla compound character recognition: A new benchmark using a novel deep learning approach," *Pattern Recognition Letters*, vol. 90, pp. 15–21, 2017, ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2017.03.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865517300703.

[8] 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, pp. 413–431, Dec. 2014. DOI: 10.1007/s10032-014-0222-y.

[9] T. C. Wei, U. U. Sheikh, and A. A.-H. A. Rahman, "Improved optical character recognition with deep neural network," in *2018 IEEE 14th International Colloquium on Signal Processing Its Applications (CSPA)*, 2018, pp. 245–249. DOI: 10.1109/CSPA.2018.8368720.

[10] N. B. Muppalaneni, "Handwritten telugu compound character prediction using convolutional neural network," in *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, 2020, pp. 1–4. DOI: 10.1109/ic-ETITE47903.2020.349.

[11] S. S. Golait and L. Malik, "Handwritten marathi compound character segmentation using minutiae detection algorithm," *Procedia Computer Science*, vol. 87, pp. 18–24, 2016, Fourth International Conference on Recent Trends in Computer Science / Engineering (ICRTCSE 2016), ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2016.05.120. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050916304598.

[12] B. Purkaystha, T. Datta, and M. S. Islam, "Bengali handwritten character recognition using deep convolutional neural network," in *2017 20th International Conference of Computer and Information Technology (ICCIT)*, 2017, pp. 1–5. DOI: 10.1109/ICCITECHN.2017.8281853.

[13] 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.

[14] M. J. Hasan, M. F. Wahid, and M. S. Alom, "Bangla compound character recognition by combining deep convolutional neural network with bidirectional long short-term memory," in *2019 4th International Conference on Electrical Information and Communication Technology (EICT)*, 2019, pp. 1–4. DOI: 10.1109/EICT48899.2019.9068817.

[15] M. R. Kibria, A. Ahmed, Z. Firdawsi, and M. A. Yousuf, "Bangla compound character recognition using support vector machine (svm) on advanced feature sets," in *2020 IEEE Region 10 Symposium (TENSYMP)*, 2020, pp. 965–968. DOI: 10.1109/TENSYMP50017.2020.9230609.

[16] A. Fardous and S. Afroge, "Handwritten isolated bangla compound character recognition," in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, 2019, pp. 1–5. DOI: 10.1109/ECACE.2019.8679258.

[17] U. Pal, T. Wakabayashi, and F. Kimura, "Handwritten bangla compound character recognition using gradient feature," in *10th International Conference on Information Technology (ICIT 2007)*, 2007, pp. 208–213. DOI: 10.1109/ICIT.2007.62.

[18] A. Ashiquzzaman, A. K. Tushar, S. Dutta, and F. Mohsin, "An efficient method for improving classification accuracy of handwritten bangla compound characters using dcnn with dropout and elu," in *2017 Third International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, 2017, pp. 147–152. DOI: 10.1109/ICRCICN.2017.8234497.

[19] A. Bagaskara and M. Suryanegara, "Evaluation of vgg-16 and vgg-19 deep learning architecture for classifying dementia people," in *2021 4th International Conference of Computer and Informatics Engineering (IC2IE)*, IEEE, 2021, pp. 1–4.

[20] 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.

[21] N. Das, R. Sarkar, S. Basu, P. K. Saha, M. Kundu, and M. Nasipuri, "Handwritten bangla character recognition using a soft computing paradigm embedded in two pass approach," *Pattern Recognition*, vol. 48, no. 6, pp. 2054–2071, 2015, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2014.12.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320314005202.

[22] M. Biswas, R. Islam, G. K. Shom, *et al.*, "Banglalekha-isolated: A multipurpose comprehensive dataset of handwritten bangla isolated characters," *Data in Brief*, vol. 12, pp. 103–107, 2017, ISSN: 2352-3409. DOI: https://doi.org/10.1016/j.dib.2017.03.035. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352340917301117.

[23] J. Ferdous, S. Karmaker, A. S. A. Rabby, and S. A. Hossain, "Matrivasha: A multipurpose comprehensive database for bangla handwritten compound characters," in *Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2020, Volume 3*, Springer, 2021, pp. 813–821.