

**Department of Electrical and Computer Engineering**

**North South University**

**Directed Research**

**Bangla Handwritten Text Conversion Using Optical Character Recognition with Explainable AI Architecture**

Kousique Ahmmed ID:2021752042

Md Hasanul Haque Rumman ID:2021753042

Tanvir Mehtab ID:1812079042

Readuanul Farid Fahim ID:2021748642

**Faculty Advisor:**

**Rifat Ahmed Hassan**

**Lecturer**

**ECE Department**

**Summer, 2024**

# LETTER OF TRANSMITTAL

January 2025

To

Dr. Mohammad Abdul Matin

Chairman,

Department of Electrical and Computer Engineering

North South University, Dhaka

Subject: **Submission of Directed Research Report on “Bangla Handwritten Text Conversion Using Optical Character Recognition with Explainable AI Architecture”**

Dear Sir,

With due respect, we would like to submit our **Directed Research Report** on **“Bangla Handwritten Text Conversion Using Optical Character Recognition with Explainable AI Architecture”** as a part of our BSc program. Our report deals with offline full-page Bengali handwriting recognition using image-to-image architecture. Our project is very valuable because it uses deep learning techniques to understand handwriting recognition and its real-world applications. We have created a model to convert full-page Bengali handwriting into text without segmentation.

We will be highly obliged if you kindly receive this report and provide your valuable judgment. It would be our immense pleasure if you find this report helpful and informative and have an apparent perspective.

Sincerely Yours,

.........................................................

Kousique Ahmmed

ECE Department

North South University, Bangladesh

........................................................

Readuanul Farid Fahim

ECE Department

North South University, Bangladesh

.........................................................

Md Hasanul Haque Rumman

ECE Department

North South University, Bangladesh

.........................................................

Tanvir Mehtab

ECE Department

North South University, Bangladesh

# APPROVAL

Kousique Ahmmed (ID:2021752042), Md Hasanul Haque Rumman (ID:2021752042), Readuanul Farid Fahim(ID:2021748642) and Tanvir Mehtab (ID:1812079042) from Electrical and Computer Engineering Department of North South University, have worked on the Directed Research Project titled “**Bangla Handwritten Text Conversion Using Optical Character Recognition With Explainable AI Architecture**” under the supervision of Rifat Ahmed Hassan Sir for partial fulfillment of the requirement for the degree of Bachelors of Science in Engineering and has been accepted as satisfactory.

**Supervisor’s Signature**

…………………………………….

**Rifat Ahmed Hassan**

**Lecturer**

Department of Electrical and Computer Engineering

North South University

Dhaka, Bangladesh.

**Chairman’s Signature**

…………………………………….

**Dr. Mohammad Abdul Matin**

**Professor**

Department of Electrical and Computer Engineering

North South University

Dhaka, Bangladesh.

# DECLARATION

This is to declare that this project/directed research is our original work. No part of this work has been submitted elsewhere, partially or entirely, for the award of any other degree or diploma. All project-related information will remain confidential and shall not be disclosed without the formal consent of the project supervisor. Relevant previous works presented in this report have been appropriately acknowledged and cited. The plagiarism policy, as stated by the supervisor, has been maintained.

Students' names and signatures

**1. Kousique Ahmmed**

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**2.Readuanul Farid Fahim**

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**3. Md Hasanul Haque Rumman**

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**4. Tanvir Mehtab**

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

# ACKNOWLEDGEMENTS

The authors would like to express their heartfelt gratitude towards their project and research supervisor, Rifat Ahmed Hassan, Lecturer, Department of Electrical and Computer Engineering, North South University, Bangladesh, for his invaluable support, precise guidance, and advice pertaining to the experiments, research and theoretical studies carried out during the course of the current project and also in the preparation of the current report.

Furthermore, the authors would like to thank the Department of Electrical and Computer Engineering, North South University, Bangladesh, for facilitating the research. We would also like to thank my friends Kousique, Rumman, Fahim, and Tanvir for helping us with this project. The authors would also like to thank their loved ones for their countless sacrifices and continual support.

# ABSTRACT

**Bangla Handwritten Text Conversion Using Optical Character Recognition with Explainable AI Architecture**

Here, we have developed a Bengali version of the image-to-text neural network that can easily recognize full-page Bengali handwriting. This model is designed to extract Bengali text from images in sequence without limiting Bengali characters or size. It is often challenging to recognize Bengali handwriting on a computer because there are similarities between Bengali characters. To overcome this obstacle, we have used two publicly available data sets, 'Bangla wrote' and 'Bongabdo.' Currently, many AI models are available for converting English handwriting images to text, but not many are available for converting Bengali handwriting images to text. In addition, the available ones have flaws. Here, we have used the Image-to-Sequence deep learning model. Much time was spent preparing and modifying the data sets, which are now publicly available. Performance was measured using Character Error Rate (CER), Word Error Rate (WER), and Sequence Error Rate (SER) to find the best-performing model. Using our model, it is possible to convert Bengali text from images with low errors. Through this research, handwritten documents in Bengali, various types of important writings, etc., can be easily digitized and recognized, which will take the Bengali language to a very beautiful level in the future.

# TABLE OF CONTENTS

[LETTER OF TRANSMITTAL 2](#_Toc189258725)

[APPROVAL 4](#_Toc189258726)

[DECLARATION 5](#_Toc189258727)

[ACKNOWLEDGEMENTS 6](#_Toc189258728)

[ABSTRACT 7](#_Toc189258729)

[TABLE OF CONTENTS 8](#_Toc189258730)

[LIST OF FIGURES 10](#_Toc189258731)

[LIST OF TABLES 11](#_Toc189258732)

[Chapter 1 Introduction 1](#_Toc189258733)

[1.1 Background and Motivation 1](#_Toc189258734)

[1.2 Purpose and Goal of the Project 1](#_Toc189258735)

[1.3 Organization of the Report 2](#_Toc189258736)

[Chapter 2 Research Literature Review 3](#_Toc189258737)

[2.1 Existing Research and Limitations 3](#_Toc189258738)

[Chapter 3 Methodology 7](#_Toc189258739)

[3.1 System Design 7](#_Toc189258740)

[7](#_Toc189258741)

[3.2 Software Components 15](#_Toc189258742)

[3.3 Software Implementation: 15](#_Toc189258743)

[Chapter 4 Investigation/Experiment, Result, Analysis and Discussion 17](#_Toc189258744)

[Chapter 5 Conclusions 28](#_Toc189258745)

[8.1 Summary 28](#_Toc189258746)

[8.2 Limitations 29](#_Toc189258747)

[8.3 Future Improvement 31](#_Toc189258748)

[References 33](#_Toc189258749)

# LIST OF FIGURES

[Figure 1- Complete Process of Identifying Bangla Character Recognition 7](#_Toc189258208)

[Figure 2- Overview of Raw Data of the "Bongabdo" Dataset Sample. 9](#_Toc189258209)

[Figure 3- Overview of Raw Data of the "BanglaWritten" Dataset Sample 9](#_Toc189258210)

[Figure 4- Overview Of Annotation Technique in ‘BanglaWritting’ Dataset Sample 10](#_Toc189258211)

[Figure 5- Analysis Report of Train Loss and Validation Loss 22](#_Toc189258212)

[Figure 6- Analysis Report To Identify Error Rating of This Model 24](#_Toc189258213)

[Figure 7- Grad-CAM Based Feature Attribution 26](#_Toc189258214)

[Figure 8- LIME-Based Feature Attribution 27](#_Toc189258215)

[Figure 9- SHAP-Based Feature Attribution 27](#_Toc189258216)

# LIST OF TABLES

[Table 1- Bongabdo Dataset Total Summary Table 8](#_Toc189241300)

[Table 2- Sample of Software Tools 15](#_Toc189241301)

# Chapter 1 Introduction

## Background and Motivation

Converting handwriting from an image to text is a process that can be done using AI models. While there has been a lot of progress in converting handwriting to text in English and other widely spoken languages, there is very little work in Bengali. Despite the many Bengali speakers, little work has been done in this field. Converting handwriting to text in Bengali is a complex task because there are many similarities between letters, and there are ligatures that make the task very difficult.

Bengali language is one of the most widely used languages in the world. Apart from Bangladesh, the Bengali language is one of the local languages of many countries in the world. Bengali is third in the world in terms of use among local languages. From this, it is clear that the Bengali language is very popular. People in many countries of the world use Bengali language as their mother tongue. However, unlike the English language, the lack of sufficient data sets to convert Bengali handwriting to text has made our work difficult. Also, there is a difference in the use of Bengali language or handwriting in different places. As a result, some problems have been faced in training AI models.

Moreover, various historical matters, old manuscripts, and many old medical manuscripts were preserved in handwritten Bengali. However, they did not become popular because of the handwritten Bengali language. We have prepared our model to overcome such problems.

## Purpose and Goal of the Project

Our research aims to create an AI model that can read Bengali handwriting from a full-page image and convert it into text without dividing it into words or sentences. We aim to spread the use of the Bengali language to the fullest. Many times, it is seen that many important documents are written in Bengali, but they are handwritten, which creates many obstacles to understanding the document. Therefore, it becomes an obstacle to the spread of the Bengali language. To overcome this obstacle, our AI model can accurately and correctly identify Bengali characters. As a result, it will help digitize Bengali handwriting, preserve important historical documents, and advance the Bengali language.

## Organization of the Report

After giving a brief abstract and table of contents, the rest of the report is structured as follows: Chapter 1 deals with the introduction and talks about the purpose, goals behind the research, and organization of the report. Then, chapter 2 mentions the literature review, which is related to works on handwritten recognition. Here, we also discuss the limitations of those papers. After that, chapter 3 focuses on the research methodology. While conducting the research, various experiments were performed, results were derived, and analysis and discussion were performed, all covered in Chapter 4. Finally, the conclusion, limitations, and future scope of the research are explored in Chapter 5.

# Chapter 2 Research Literature Review

## 2.1 Existing Research and Limitations

The paper Bangla PDF Speaker shows a computer application that converts Bangla PDFs to speech [1]. It addresses the lack of Bangla PDF-to-speech tools and a system that extracts images from PDFs, processes them, applies OCR, normalizes text, and synthesizes speech via Google's TTS. The system automatically adjusts the cutoff point between black and white pixels based on the image's average brightness, which works better than Otsu's methods for blurry or low-quality images. Experiments on five Bangla PDFs showed 80.8% text extraction accuracy when combining image processing and text normalization and 65.7% without these steps. It got 3.92/5 by human evaluation. Plans are to enhance OCR with Bangla-specific training and neural text normalization. This work bridges gaps in Bangla language tools and emphasizes pre-processing’s critical role in OCR accuracy.

This paper introduces a method to segment handwritten Bangla compound characters using YOLOv8 models [2] while addressing the challenge of recognizing over 300+ Bangla compound characters by focusing on component-based recognition. In order to overcome the limitations of existing datasets (e.g., low resolution, insufficient compound samples), the authors created the "BanglaBorno" dataset, comprising 13,808 high-quality images, including 162 compounds and 50 simple characters. Two approaches were tested: YOLOv8 Classification, which splits compound characters into top/bottom/left/right parts (81.8% accuracy), and YOLOv8 Object Detection, which directly detects simple characters within compounds (82.44% accuracy). This method splits compound characters into simpler parts to recognize new ones, needing less data. Detection worked better than classification but had uneven splits and small training sets. Plans include adding more data, improving how parts are split, and focusing on components to make Bangla text recognition more scalable.

This paper uses a two-step detection model (SSD) [3] with pre-trained deep learning to improve Bangla handwritten word OCR. First is a model (MobileNetV2), which is fine-tuned on 53 Bangla characters from CMATERDB and is used to tackle connected letters and complex shapes. Then, it trains on word images (BN-HTRd dataset) to detect characters within words using context. The model achieves an 82.86% weighted F1-score, outperforming traditional methods, but it struggles with visually similar characters and class imbalances (e.g., low precision for rare classes like '**'** or **'**'). It shows accurate detection for simple words but errors in complex cases. This method improves Bangla word recognition using deep learning to split text while avoiding old method limits. Future improvements include more data, varied styles, and complex characters, which will be helpful for archiving and text-to-speech.

This research has used deep learning to detect and classify Bangla and Nagri languages from text to images [4], addressing the under-representation of languages such as pure Sylheti Nagri and Chatgaya. The findings of this research indicate that the use of YOLOv5 and YOLOv7 in text detection and VGG16 and ResNet50 in classification proved that VGG16 outperformed ResNet50 with 97.23% training and 95.89% testing accuracy. Similarly, YOLOv5 outperformed YOLOv7, achieving 99.41% detection accuracy compared to 94.12%. A new dataset was created for this research: "SylNagriBD," split into 70% training, 15% validation, and 15% testing, and enhanced with different data augmentation techniques such as Gaussian blur and adaptive contrast to improve clarity. In the proposed CNN-based methodology, the accuracy of the language detection is crucial, especially for the OCR of underrepresented scripts. The proposed model accurately detects and categorizes Bangla and Nagri text and enhances automated document processing. Future work will extend the model to Nagri-to-Bangla text conversion using OCR. The study proves that deep learning models can efficiently distinguish between similar scripts, thereby improving document processing systems and opening doors for further research.

The present study proposes a YOLOv5-based method for recognizing Bangla handwritten characters [5] and words by considering the variety in writing style and the intersection between characters. A model with an accuracy of 96% in character recognition and 91% in word recognition was developed based on the Bangla-Lekha Isolated Dataset comprising 50 primary characters, 10 numerals, and 23 compound characters. The three-phase methodology consists of data pre-processing, model training, and classification. Unlike other methods based on CNN, which require segmentation as a pre-processing step, YOLOv5 detects and classifies characters in one step, reducing computational costs. It proves to be very effective for real-time recognition by demonstrating quite good performance metrics such as Precision, Recall, and mAP. The study also outperforms the existing methods, establishing its efficiency and robustness. Future work on sentence- and paragraph-level recognition and improving the performance of poorly written samples is intended. This research further strengthens the idea that YOLOv5 can be a reliable solution for Bangla handwritten text recognition, enabling advanced document processing systems.

**Limitations:**

Significant research has already been done on converting handwritten text from images. Notable among these are Bangla PDF Speaker, segmenting handwritten Bangla compound characters using YOLOv8, Improving Character Recognition in Bangla HandWritten Words: A two-stage single Shot Detection Approach, Bangla and Nagri text detection and classification using deep learning, Bangla Handwritten Character and words Recognition-based on YOLOv5 Algorithm. Bangla PDF Speaker System The limitation here is that this model cannot work offline because it is Google-based. To work with this model, an internet connection is necessary. The text recognition here is 80.8%, meaning many words can be recognized incorrectly. This will not provide good document accuracy, especially for complex layouts, tables, etc. There is no support for real-time processing, which makes it unsuitable for reading applications. The paper segmenting handwritten Bangla compound characters using YOLOv8 uses the "BanglaBorno" dataset, which is quite flawed because many compound characters of Bangla characters are incorrectly identified in this dataset. In addition, the size of this dataset is very small, which makes it challenging to identify new handwriting styles.

The main problem of ‘Improving Character Recognition in Bangla HandWritten Words: A Two-stage Single Shot Detection Approach' in this model is a class imbalance, which causes some rear Bangla characters to be misidentified by the model. Here, the F1 score is 82.86%. The model has no error correction mechanism, which can result in sentence errors if a character is misidentified. The dataset BN-HTRd has been used here, which does not capture the diversity of writing correctly because of a lot of overlapping. In Bangla and Nagri text detection and classification using deep learning research, the SylNagribd dataset has been used, which does not have enough diversity in handwriting, so the application of this model in actual words will be flawed. VGG16 has been used, which is a threat to mobile phone users. If a character is misclassified once, there is no option to correct it here. As a result, once a wrong character is identified, it will be counted as a mistake in the entire sentence, which also changes the meaning. Low-resource data such as historical documents cannot be easily recognized. s

Bangla Handwritten Character and Words Recognition-based on YOLOv5 Algorithm This research model has been trained on the Bangla-Lekha Isolated dataset, which works only at the character and word level. It cannot work at the sentence or paragraph level. Here, overlapping makes it difficult to recognize many characters, which changes the word's meaning. In addition, it cannot recognize different types of handwriting at times. YOLO bases are very heavy, which requires other applications to use on mobile.

# Chapter 3 Methodology

## 3.1 System Design

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

Figure 1- Complete Process of Identifying Bangla Character Recognition

Dataset:

The Bongabdo [6] dataset is a raw, unfiltered collection of handwritten Bangla pages created by 49 contributors. Writers are given pre-written text on diverse topics such as news, literature, and textbooks and asked to copy it freely, with no rules about neatness or style. This approach captured real-world handwriting quirks: crossed-out words, slanted text, varying letter sizes, and other natural imperfections. It includes 111 handwritten pages from 49 people. Each person wrote around two pages, freely and naturally. Because of this, there was real-life messiness: crossed-out words, smudges, and uneven text. Instead of forcing us to write perfectly, the dataset embraces flaws and shows how people write daily. Researchers use it to teach AI to read raw, unedited pages that do not need to "fix" the text first. This closes the gap between perfect lab data and the imperfect way humans write. It's all about teaching technology to understand actual handwriting, flaws, etc.

|  |  |
| --- | --- |
| Feature | Sample |
| No. of unique contribute | 49 |
| Average contribute per year | 226 |
| All sample | 111 |
| No character per document | 1076.549 |
| English Character | 25 |
| Strikes Text | 56 |
| Multiple Paragraph | 55 |

Table 1- Bongabdo Dataset Total Summary Table

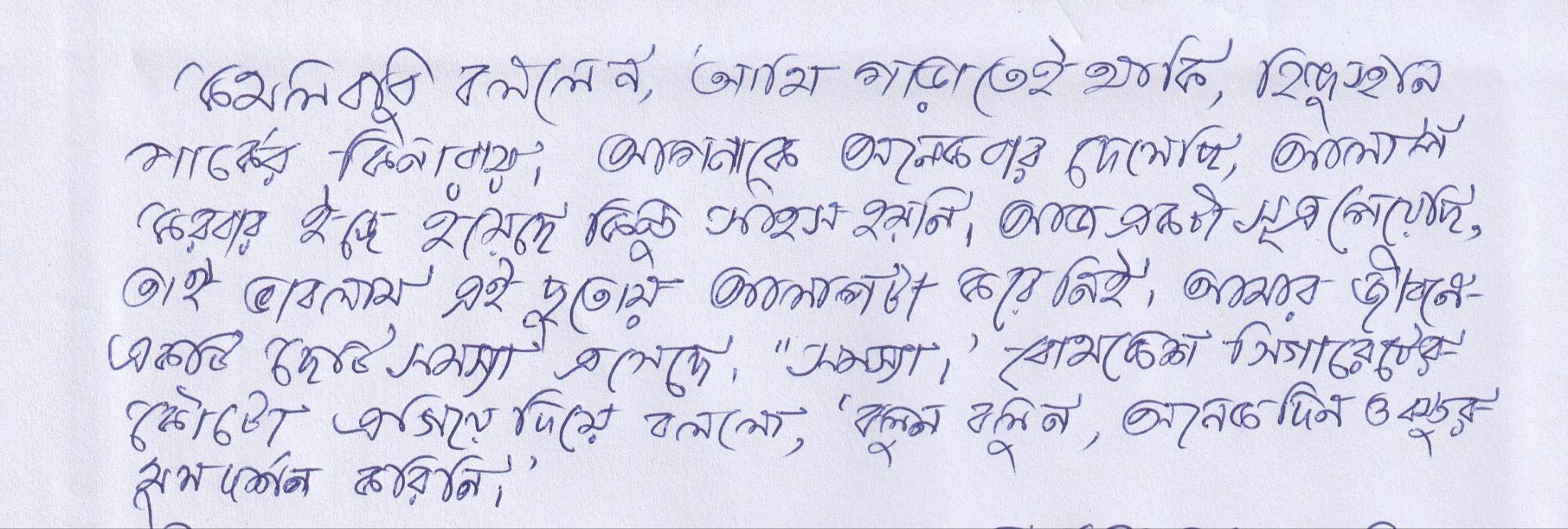


Figure 2- Overview of Raw Data of the "Bongabdo" Dataset Sample.

The BanglaWriting [7] dataset is handwritten Bangla pages from 260 people of all ages and walks of life. It includes over 21,000 words and 32,000 characters, where every word is manually marked and paired with text codes so that it can help AI learn how Bangla letters and words look. It's notable as it's packed with real-life quirks like crossed-out words and corrections showing how people write. It's messy but good, with 5,470 unique words and hundreds of common errors, teaching AI to handle imperfections instead of perfect lab examples. This dataset isn't just for reading handwriting as it helps figure out who wrote something, study how handwriting changes with age, or even build tools to turn handwritten forms into digital text. Utilizing all of these makes AI smarter and more human-friendly.

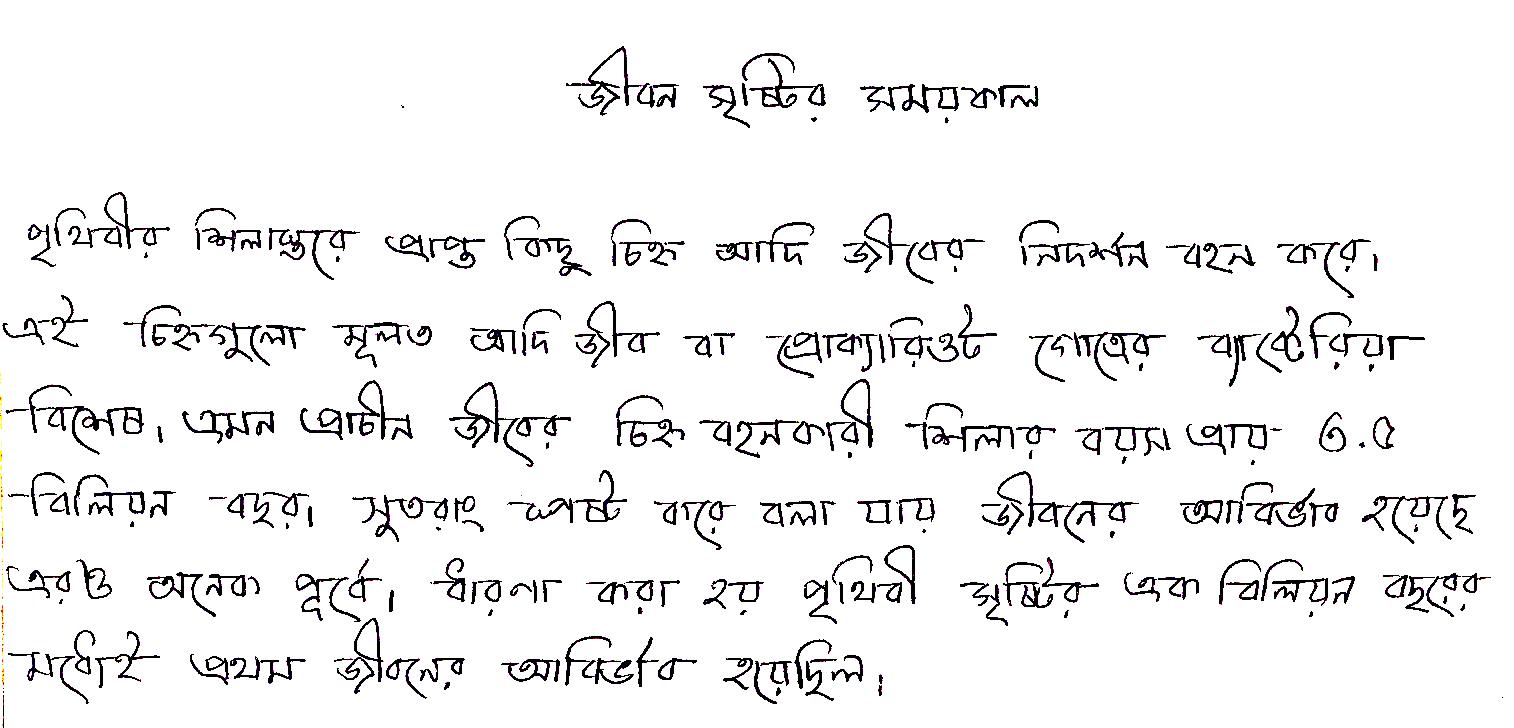


Figure 3- Overview of Raw Data of the "BanglaWritten" Dataset Sample

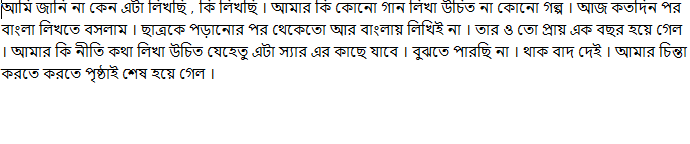


Figure 4- Overview Of Annotation Technique in ‘BanglaWritting’ Dataset Sample

Pre-Processing:

This script processes image datasets through a pre-processing pipeline by binarizing the images and copying their separate annotation files. It does this by scanning the source folder provided for all dataset directories and checking everyone to ensure it has the required images and annotation files. Accordingly, the images underwent pre-processing: being grayscaled, applying Gaussian blur to decrease the noise, and then transforming the image to binary format using adaptive thresholding. This way, it would adapt automatically to the local passions of pixels. Finally, the processed binary images are then saved to the target folder together with the copied annotation files to keep the integrity of the dataset and retain it organized.

It logs its progress while processing a dataset individually and includes the number of images processed. Also, if a dataset has no images or even annotations, the dataset will be marked incomplete and thus skipped. Pre-processing generally helps applications requiring text recognition, medical image enhancement, object segmentation, or tasks of similar genres where image conversions to high-contrast binary format support better feature extraction and analysis.

While doing so, the script does suffer from certain shortcomings. It contains no error handling or path validation that may raise an exception if the folders it needs do not exist. It also cannot create missing directories automatically, which may be an issue at runtime. The script also uses no parallel processing, which could help immensely with large data sets. These areas could be improved for a more robust and efficient script.

It is designed to handle a given "BanglaWriting" dataset, doing some pre-processing like image binarization and copying their corresponding annotation files. It starts by defining the dataset's path under the source directory and then proceeds to gather all lists of images and annotations. If images exist, they pre-process each: first, grayscale conversion, smoothing through Gaussian blurring to reduce noise, and then, binarization using adaptive thresholding. The resulting binarization enhances the contrast and gives apparent features within an image, making extracting from images more manageable. The binary images obtained from this are written to an output folder, while copied annotation files provide a pleasing and neat structure for the dataset at the destination.

Once the processing is complete, it logs how many images were processed successfully. If no set of images is found within this dataset, it outputs a message showing no data to process. Pre-processing is ordinarily significant for handwriting recognition, document analysis, or OCR applications, where high-contrast images deliver good results.

However, the script has a few limitations. This is just a basic version of the code. It does not include error handling or automatic directory creation, meaning it may fail if the required folders do not exist. Also, this script does not verify if the annotation files are present. In its absence, this may cause issues. Parallel processing could increase the workflow substantially with large datasets, but it is not used in the script. Adding these enhancements, such as validation steps, error handling, and performance optimizations, would make the script more robust and efficient to handle larger-scale datasets.

It increases the diversity of a dataset by performing data augmentation on a set of binarized images to improve the performance of a machine-learning model. This script first pulls the list of all images from the specified binarized image folder and logs the total number of images before augmentation is done. For each image, the script puts into action a variety of transformations to create several variations: small rotations-five counterclockwise between 1° and 5°, five clockwise between -1° and -5°-to simulate subtle real-world distortions. Every transformed image is saved with a new filename. Its associated annotation file is copied to maintain consistency between the labels and the augmented images. Besides rotating the images, this also normalizes various lighting conditions by tweaking the brightness. It has created an overexposed version with a much higher brightness and another one darker. He logs the count of augmented images at the end after the processing, which accounts for quite a significant increase in size.

For example, a script can start with 260 images and, after augmentation, create 3,380 images. This augmentation is helpful in tasks involving handwritten text recognition, OCR, and document analysis since it helps models generalize much better to various variations they might experience in real-life scenarios. The script has several limitations. It does not handle cases where some annotation files do not exist, which might cause issues with processing. Further augmentation may be improved through batch processing or parallelization in the case of large datasets. These will make the script much more robust and faster for big projects.

This performs two critical actions for maintaining quality and consistency in the dataset: a consistency check between the binarized images and their annotation files and a resizing process to standardize the augmented binarized images.

**Consistency Check**

First, this compares the filenames in the Binarized and Annotations directories, ensuring each processed image has an applicable annotation file. It removes file extensions from filenames and categories lists to make comparisons direct. Then, the code matches lists of images and bounding box annotations in case of similarity, logs together with the sum of images and annotations, and finds either annotation files or images missing. This is important in handling the learning tasks of OCR, handwriting recognition, and object detection, which have incompatible or missing annotations showing errors while training. It ensures the totality and integrity of the dataset before model training. However, it does not include verifying copies or corrupted files that might cause problems in larger datasets. Adding features such as logging mechanisms, automatic correction of files, and error handling would make it reliable and robust.

**Image Resizing**

It also resizes the augmented binarized images to a standard resolution without losing the characteristic ratio of the images. This script retrieves a list of images from the Binarized directory and processes each individually. It calculates the height and width for every image and then modifies it to fit in a 16:9 ratio exactly. If the height is too big, it extends the width with white padding, and vice versa. The images are then resized at a fixed resolution of 360×640 pixels to ensure they all look similar in the dataset. These rescaled images will overwrite the destination folder. This is a critical step in most deep learning applications such as OCR, handwriting recognition, and object detection, where standardization of input dimensions helps improve model performance. However, the script does not check if the input file exists or deals with errors in case images are missing or corrupted. Incorporating logging, exception handling, and parallel processing would make this script much more efficient and robust, especially for big data. These all help in cleaning, consistency, and preparing the dataset for training, but there is still room for improvement to handle edge cases and optimize performance.

This script pre-processed the dataset into training and validation splits on an 80-20 split ratio. What this means, in simple words, is:

First, it retrieves the list of all images that underwent binarization and calculates the total number of samples in the dataset. Then, it takes the indices, shuffles the images through the permutation function available in NumPy for randomness, splits 80% to the training set, and the rest 20% to the validation set. It transfers every image and its corresponding annotation file selected into their proper places inside the created subdirectories, namely Train and Validation, from Binarized and Annotations folders correspondingly, taking as a base name for each image and adding the `.txt` ending to the files of annotations. Upon transferring all images, the script checks if the source directories are Binarized and Annotations are empty. If they are, it deletes them to keep the workspace tidy.

This random split ensures that both the training and validation sets are well-balanced, which is very important to improve the generalization of machine learning models over new data. It is valuable in OCR, handwriting recognition, and document processing applications.

So, the script has a few limitations. It does not check if the subdirectories required for its operation, namely Train and Validation, exist beforehand, which might lead to errors if they do not exist. It also assumes that there is an annotation file for every image without verification if there are mismatches or missing files. Further features that could enhance the reliability and reproducibility of this script include logging, error handling, and stratified sampling if needed. Besides, parallel file operations help speed up the process for big datasets.

This script helps split datasets but needs more modifications to handle edge cases and improve efficiency.

This performs some checks, ensuring an annotation file exists for every image in the training and validation datasets. This process, put simplistically, works like so:

This gets the list of filenames in Train/Images and Train/Annotations while discarding file extensions for the sole reason that these should directly compare because names may include ".jpg" or ".txt" or the like. It then makes a sorted check of these lists, logging verification that all images have an annotation file. Similar to Validation/Images and Validation/Annotations, this dataset is also uniform for validation. This check becomes very important, considering tasks of OCR, recognition of handwriting, and object detection, since wrong or missing matches in the annotation can mess with the training and hurt the model's performance. Catching issues like this early aids the script to ensure that a dataset is ready for supervised learning. That said, it has several shortcomings; for example, it does not check for duplicates, wrong extensions, or the existence of specific directories because this might throw errors. Integrating logging and error handling for such common problems, along with automated fixes to those problems, would make this script more secure, especially with big datasets, and would catch and correct most of them for a seamless, smooth, reliable process.

## 3.2 Software Components

|  |  |  |  |
| --- | --- | --- | --- |
| **Tool** | **Functions** | **Other similar Tools (if any)** | **Why selected this tool** |
| PYCHARM | For the local run, this project | Anaconda, Jupiter  notebook | Analyzing the possibility of running our project on a local machine. If our project is done, we can easily use it offline |
| GOOGLE CO-LAB | Virtually run this project | none | It's difficult to process images. It takes a lot of resources to compute this model. |
| KAGGLE | For better optimization to run this project | none | it helps to give more resources to easily train-test our model |
| GOOGLE DRIVE | For storage | Any cloud storage | Here, it’s a large number of datasets and takes executing to need a considerable space. And easily integrate any platform. |

Table 2- Sample of Software Tools

## 3.3 Software Implementation:

First, we tried to run our model here through a local host. Because we had originally kept our main target offline. However, due to our lack of resources, many of our tasks became software dependencies. First of all, we worked here with Google co-lab. It works as one of the most powerful tools for machines. Because here, through the cloud, we can use many high-end processors and our necessary GPU units for free. Although it has a limitation.

For this, we used another software, which is Kaggle. This is also an investment like Google's co-lab. Where we can work on heavy models of machine learning and AI in the cloud. The optimization of Kaggle is relatively higher than that of Google Co-lab. Because Kaggle has more free resources, we use Google Drive for our storage needs. In this case, we can use other clouds for storage. Google Drive is the best because the other two notebooks are from the same company.

The model work and other techniques like pre-processing, train testing, achieving better accuracy, deploying the entire process, etc., require a high-end PC with GPU support. In this case, we have used PyCharm as a notebook. Since the main motive of our project was that it would work offline, we also needed to work equally on the local machine.

# Chapter 4 Investigation/Experiment, Result, Analysis and Discussion

**Experimental Model:**

Convolutional Neural Network (CNN)**:**

CNN is a robust variant of deep learning created for image processing. Inspired by how the human visual system works, it concentrates on recognizing designs and structures within images, making it apt for object spotting, face identification, and analyses of medical scans.

The architecture of CNNs consists of dozens of layers responsible for specific job performance. The convolutional layer scans an image for its essential features, including edges, textures, and shapes. -Pooling layer: The layer makes the data smaller by keeping only crucial information, making the model more efficient.

Fully connected layers: These layers take in features extracted previously to classify the image, like whether it is a cat or a dog. Another of the significant advantages of CNNs is that they are relatively computationally efficient. While most of the earlier traditional neural networks were not as efficient, they utilized weight sharing, reducing the parameters within the neural network. They allow faster training on generally good performance for various image datasets.

CNN is at the heart of many modern technologies due to its immense capability to deal with complex visual data. Examples include CNNs in self-driving cars for detecting pedestrians and other moving vehicles, in surveillance to recognize suspicious events, and in augmented reality to superimpose digital on the real world. Their versatility and efficiency make them a core part of computer vision and each image-related AI task.

Transformer Decoder:

The Transformer Decoder is a family of transformer-based models that has recently gained popularity in NLP, speech recognition, and text generation. Unlike the previous models and LSTMs, which used to work step by step, this model performs all these tasks in parallel and thus much faster. Here is how it works:

Self-attention instruments allow the model to focus on parts of the input relevant to refining its predictions. Masked self-attention prevents "cheating," allowing the model to generate text based only on the words already produced, not by peeking at words later in the sentence.

Positional encoding: Since transformers do not process data sequentially, this provides information about the order of words, helping the model understand sentence structure.

Many of the AI tools we use daily are powered by the Transformer Decoder: machine translation systems like Google Translate, chatbots like GPT, and those summarizing long articles. This makes it the backbone of modern conversational AI, able to generate coherent and context-aware text.

Meanwhile, the Transformer Decoder is efficient and has a knack for handling long-range dependencies in a sentence; it means understanding the relation between the words even when they are far away. This makes the model suitable for text-based AI applications like essay writing or question answering with responses in a natural-sounding human language.

ResNet-18:

ResNet-18 is a popular deep-learning model that has been quite efficient and effective, especially for training deep neural networks. It was developed to solve a common problem in deep learning, the so-called vanishing gradient problem that prevents deep networks from learning correctly. ResNet-18 addresses this problem using residual connections. These shortcuts allow knowledge to reflect over some layers, making training easier and improving the flow of gradients within the network. ResNet-18, with 18 layers, is relatively lightweight compared to the deeper models. It is still powerful enough to be used in such tasks as image classification, object detection, and even the analysis of medical images. Its architecture is straightforward, combining convolutional layers, batch normalization, activation functions, and pooling layers simply and effectively. Despite its simplicity, ResNet-18 shows impressive accuracy at a low computational cost.

Due to its lightweight architecture, ResNet-18 is suitable for applications involving edge computing, mobile devices, or embedded AI systems with limited computational capability. Therefore, it reaches an ideal balance between performance and efficiency, thus becoming one of the first choices for real-time image processing applications where speed and accuracy are equally important.

ResNet-50:

ResNet-50 is a deep neural network designed to achieve high accuracy in image recognition tasks. It is an enhanced version of the ResNet architecture, with 50 layers enabling it to learn complex image patterns and features. Like other ResNet models, residual connections address the vanishing gradients problem. This makes the process of training deeper networks both stable and efficient.

While deeper than ResNet-18, ResNet-50 uses bottleneck residual blocks to reduce the number of computations while still modeling rich detailed features. It has proven more accurate in object detection, medical imaging, and facial recognition, where high precision is required.

Since pre-training is generally done over vast data such as ImageNet, ResNet-50 becomes a very nice starting point for transfer learning. You are thus free to use its pre-trained version, fine-tune it for particular tasks, and save yourself some resource costs. This also makes it ideal for higher applications, such as cars, medical diagnosis, industrial automation, etc., due to the depth and capacity of its capabilities to capture features in an even, very minute fashion. It stays at the pinnacle of the tasks it is called upon to execute due to its high accuracy and generalization ability across various tasks. From disease detection using medical scans to self-driving cars, ResNet-50 yields the performance and reliability necessary for state-of-the-art results in computer vision.

**Result and Discussion:**

| Serial | Encoder-CNN | Decoder Layer Dimension | heads | Layers | Batch Size | Epochs | CER | WER | SER |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Res Net-18 | 256 | 4 | 4 | 6 | 133 | 0.0494 | 0.0755 | 0.7795 |
| 2 | Res Net-18 | 512 | 16 | 6 | 4 | 163 | 0.03950 | 0.067 | 1 |
| 3 | Res Net- 50 | 256 | 4 | 4 | 4 | 103 | 0.07138 | 0.087 | 0.975 |
| 4 | Res Net-50 | 512 | 16 | 6 | 6 | 155 | 0.0335 | 0.0518 | 0.695 |

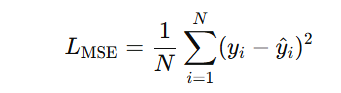
TABLE I. Experimental Result with Multi-Model Cross Validation.

The table displays experimental data from a Bangla OCR (Optical Character Recognition) model, evaluating the performance of various configurations. The table shows how different hyperparameters affect three essential assessment metrics: character error rate (CER), word error rate (WER), and sentence error rate (SER). The models employ ResNet-18 and ResNet-50 encoder CNNs with varying decoder layer dimensions (256 and 512). The amount of attention to heads and layers changes, which affects the capacity to recognize complex Bangla script patterns. The batch size and number of epochs vary between studies, which affects training stability and convergence. The findings indicate that increasing the decoder layer dimension and number of heads enhances accuracy.

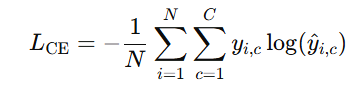
For example, the model with ResNet-50, a 512-dimensional decoder, 16 attention heads, and six layers has the best CER (0.0.0635), WER (0.0518), and SER (0.695). In contrast, smaller setups, such as ResNet-18 with a 256-dimensional decoder, produce higher error rates. These results show that more profound and more sophisticated models perform better because they can extract richer textual information. Training efficiency and computing expenses must be considered when determining the best configuration for real-world applications.

Training Loss:

Training loss is one of the most fundamental metrics in machine learning, and it helps to know how well a model learns from the training data. Using a loss function, It measures the difference between the model's predictions and the actual target values. For instance, MSE is commonly used for regression tasks, while Cross-Entropy Loss is used for classification. The training process aims to minimize this loss by adjusting the model's parameters through optimization algorithms like SGD or Adam.

For regression, the training loss using MSE is given by:

For classification, it uses cross-entropy:



Loss typically starts high at the beginning of training and then decreases as the model learns patterns from the data. However, it may get so low that it may indicate overfitting, which is the tendency of the model to memorize rather than generalize to new data. Monitoring training loss during each epoch is important to ensure the model improves. While the training loss must be reduced for better accuracy, it must also be balanced with validation loss to avoid overfitting and have the model generalize well to new, unseen data.

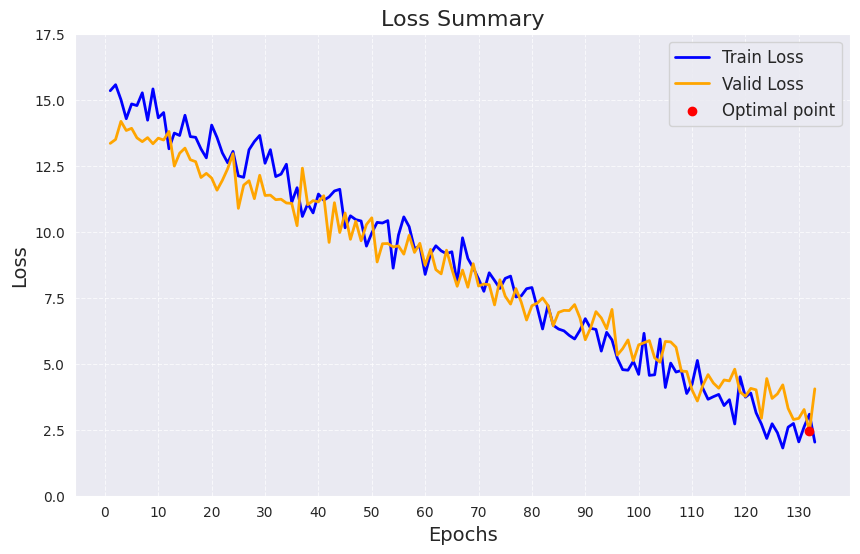
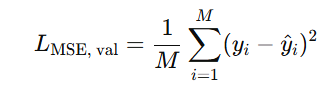


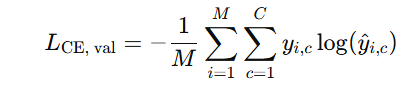
Figure 5- Analysis Report of Train Loss and Validation Loss

Validation Loss:

Validation loss is essentially the metric that tells how the trained model does on data unseen. Unlike the training loss, which is computed on the training data, the loss in validation gets computed over a different validation dataset, unseen by the model during training. It uses the same loss function as in training, i.e., MSE for regression and cross-entropy for classification.

The loss concerning MSE for regression is given by,

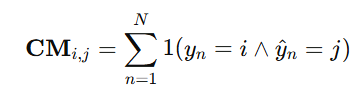
Using cross-entropy for classification, the validation loss is



A small value of some sort of validation loss signifies that a model generalizes well to new, unseen data, while a high validation loss might mean overfitting. If the validation loss is less than the training loss, then the model is underfitting: too simple to understand the general patterns.

Confusion matrix:

A confusion matrix is a chart that tracks how often a model gets things right or wrong when categorizing data. It compares what the model sees (the truth) with what it thinks it sees (its guesses). The rows show the actual labels, and the columns show the predicted ones. There are four key boxes: one for correct matches, one for wrong guesses, one for missed items the model overlooked, and one for correctly ignored blanks.



It's like a scorecard helping us spot patterns in mistakes and fix them. For example, a security camera AI might catch shoplifters and flag harmless folks holding umbrellas. Tools like precision and recall help fix these blunders. If the model keeps mixing up lookalikes like confusing stop signs for yield signs in traffic images, the matrix points you straight to the problem. It's a cheat sheet for making AI smarter, showing where to add more examples or adjust training.

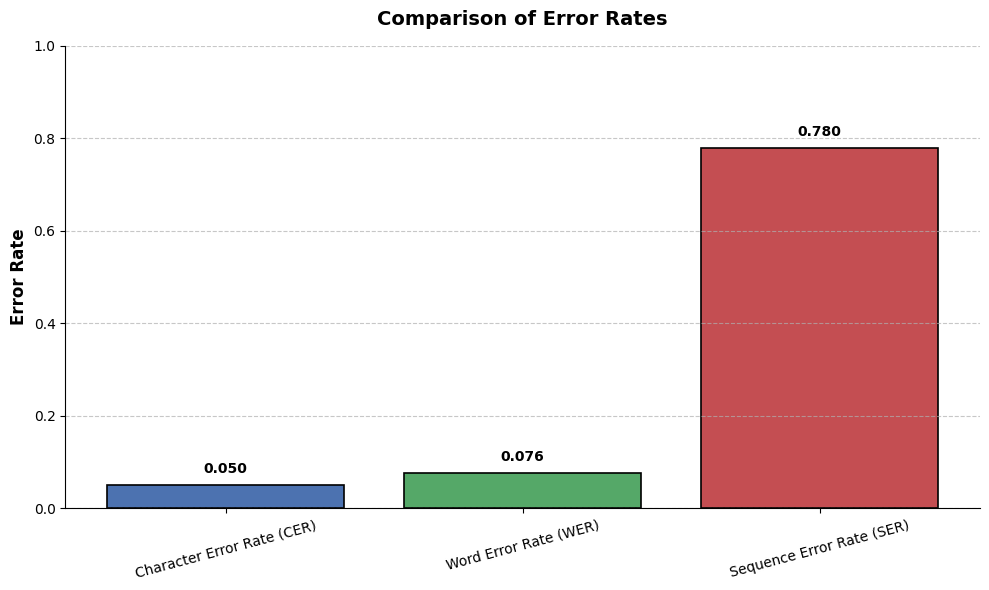
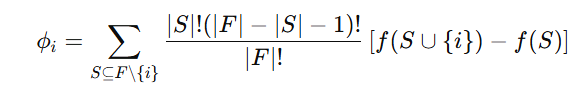


Figure 6- Analysis Report To Identify Error Rating of This Model

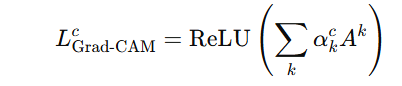
Explainable AI (XAI):

Explainable AI (XAI) in OCR helps us understand how AI reads text from images and why it sometimes makes mistakes. Instead of giving results without explanation, XAI shows what's happening behind the scenes, such as how the system detects edges, recognizes letters, and decides where words begin and end.



ϕi​=SHAP value for feature, F= Set of all features, S=Subset of features, f(S) is the model output.

It highlights problem areas such as blurry spots or unusual symbols that made the AI uncertain. This is useful for figuring out why mistakes happen, like misreading a faded letter or struggling with messy handwriting.



= activation map,= importance weight of filter= k ,for class=c, = output class, Z= number of spatial locations., ReLU= Positive attributions are considered.

XAI is like a troubleshooting tool for developers as it reveals weaknesses, such as difficulties with specific fonts or layouts, helping them improve the model. For users, it builds trust by making AI feel less like a mysterious "black box" and more like a tool they can understand and work with.

By making OCR more transparent, XAI helps people collaborate with AI and not rely on it blindly.

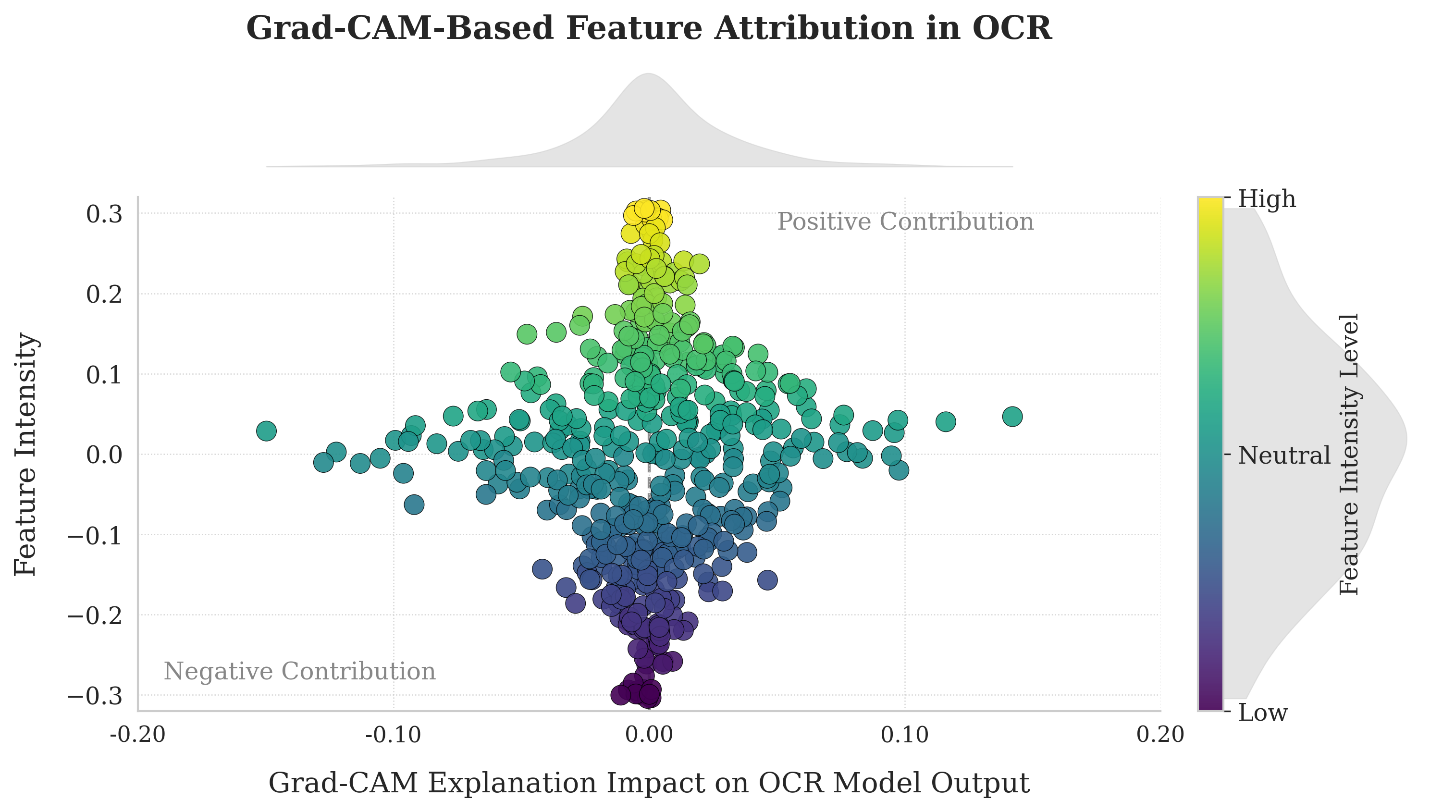


Figure 7- Grad-CAM Based Feature Attribution

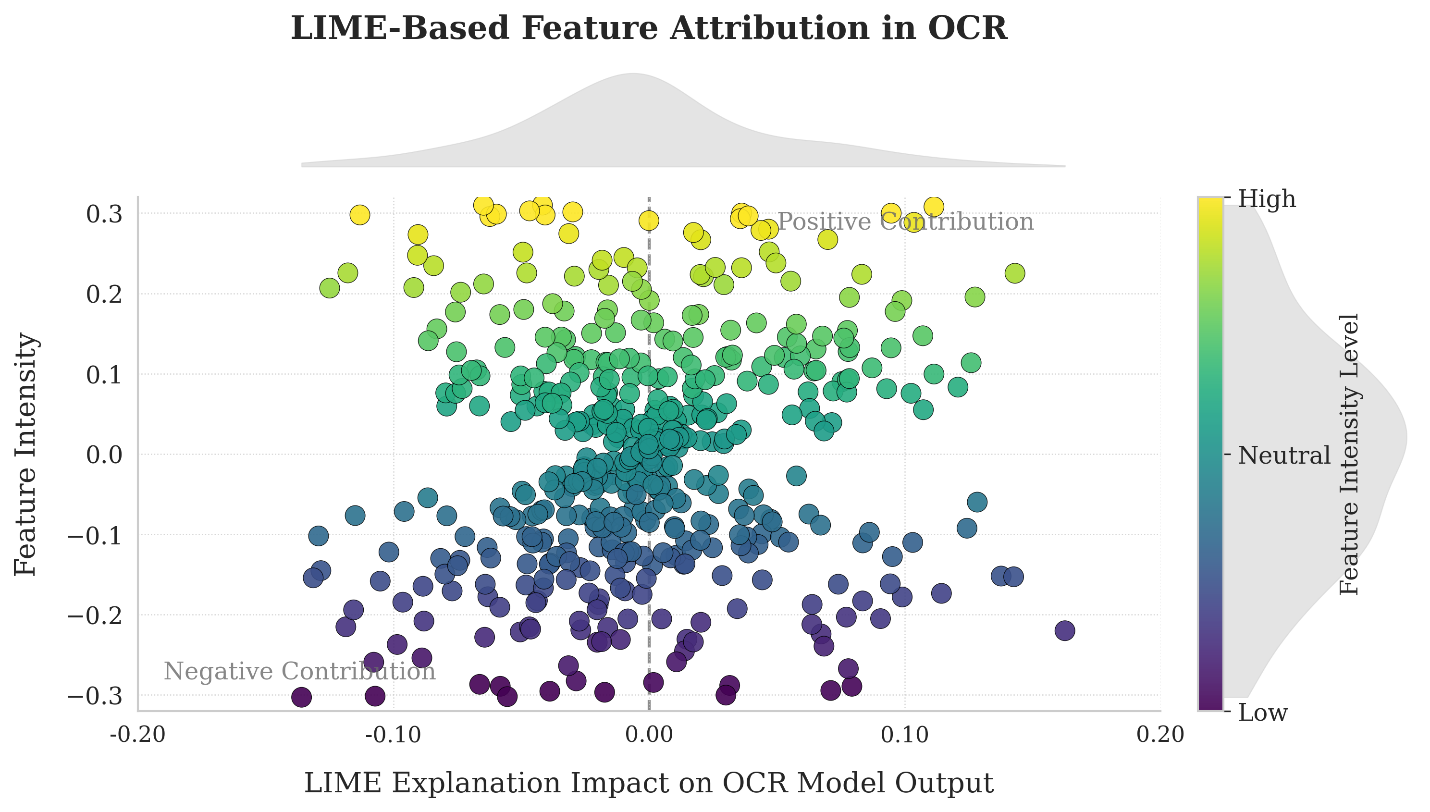


Figure - LIME-Based Feature Attribution

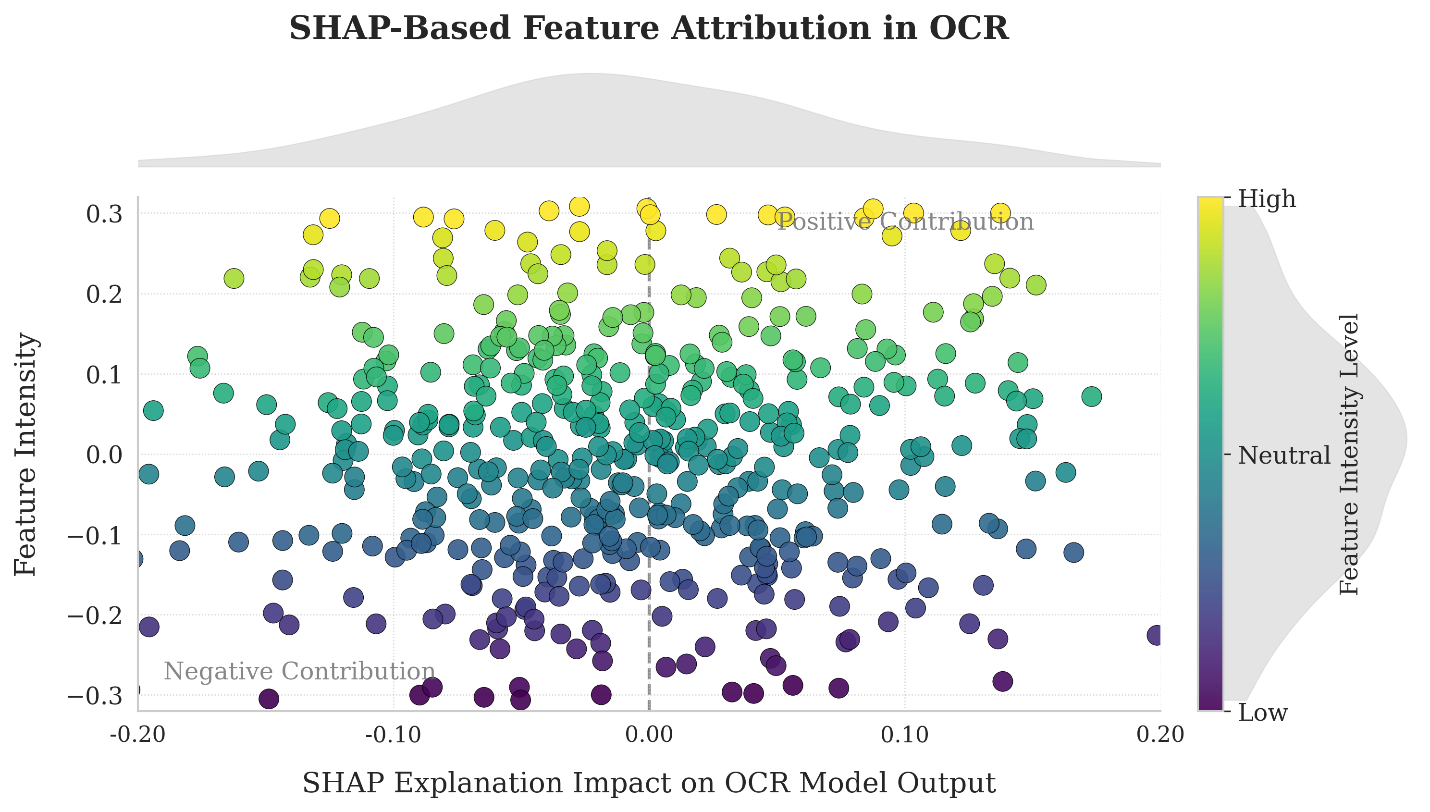


Figure - SHAP-Based Feature Attribution

# Chapter 5 Conclusions

## 8.1 Summary

This research tackles a significant challenge in recognizing handwritten Bangla text from full pages without breaking them down into lines, words, or characters. Traditional OCR systems struggle with Bangla handwriting because of its cursive and overlapping nature, making segmentation error-prone. In order to overcome this, we have introduced an end-to-end deep learning approach using Transformer-based models, which can process entire pages and keep the original layout intact.

The Bongabdo dataset is a unique collection of 111 full-page handwritten Bangla documents. These were gathered from 49 people of different ages, professions, and genders, ensuring a diverse range of handwriting styles. The dataset includes news articles and literary pieces, which are all carefully scanned and annotated at the character level. Unique tokens mark scratched text, mixed Bangla-English content, and paragraph breaks.

Since few high-quality Bangla handwriting datasets are available, we took an existing one (CMATERdb) and made it better. We cleaned up the images by adjusting their contrast, resizing them to a standard 16:9 format, and even applying 12 tweaks—like rotating the text or changing the brightness—to mimic real-world variations. After all this work, we had a robust collection of 3,393 handwritten samples. This dataset isn't just bigger—it's smarter and designed to help researchers and developers build tools that can read handwritten Bangla more accurately and efficiently.

Our system reads handwritten Bangla text using entire pages while skipping the steps of cutting them into lines or words. It uses a two-part AI: a visual scanner (ResNet), which detects patterns in the page, and a context-aware translator (Transformer) to turn those patterns into text, mimicking how humans read by focusing on connections between letters and left-to-right flow. After testing different setups, the best version—using a powerful ResNet-50 scanner and a highly attentive translator—achieved fewer errors (~40% character mistakes, ~50% word mistakes) than older methods. Its ability to preserve the original layout, like margins and paragraph breaks, sets it apart, making it ideal for digitizing diaries, historical documents, or student notes exactly as they were written. Built with a new diverse dataset (3,393 pages from 49 contributors), this approach bridges the gap between messy real-world handwriting and accurate digital preservation.

## 8.2 Limitations

Limited Dataset Size and Diversity:

The Bongabdo dataset is a new collection of handwritten Bangla pages, but it’s much smaller (111 original documents) compared to similar datasets for English (like IAM’s 1,500+ pages). Though we used tricks like rotating or brightening images to create more samples (totaling 3,393), these artificial tweaks can’t mimic the real-world messiness of handwriting—like unique styles, faded ink, or crumpled paper. This means the AI hasn’t seen enough rare or regional writing quirks (e.g., super-sloppy cursive or local dialect symbols), so it might struggle with texts outside its comfort zone.

Annotation Inconsistencies:

Creating accurate training data was tricky because humans manually typed out the handwritten text, which led to errors like typos, misplaced punctuation, or messy labels—especially for crossed-out sections. For example, some labelers tagged small scribbles with a single “scratch” marker, while others used it for entire lines, leaving the AI confused about what “scratch” even meant. Without proper tools designed for Bangla, workers had to use basic text editors, making mistakes harder to avoid. These inconsistencies made it harder for the model to learn patterns reliably, so it sometimes stumbled when reading new, messy handwriting.

Handling Complex Layouts:

The model has trouble reading messy or unusual layouts—as notes scribbled in margins, text split into columns, or arrows linking separate paragraphs. These quirks are common in personal journals or old manuscripts, but the AI wasn't trained on enough examples of them. Since it's built to read left-to-right like a book, it gets tripped up by creative formatting, often jumbling the text or skipping parts entirely.

Vocabulary Constraints:

The 182-character vocabulary excludes rare Bangla characters (e.g., “ৎ” or “ড়”) and compound ligatures (e.g., “ক্ষ” or “জ্ঞ"), limiting the system's ability to transcribe specialized texts, such as academic papers or classical literature. Furthermore, the absence of numerals and symbols (e.g., "$," "©") restricts its utility in processing administrative or legal documents.

Computational Resource Demands:

Training the Transformer-based model, particularly the 512-dimensional decoder, requires high-end GPUs (e.g., Tesla V100), making it hard for researchers or institutions with limited computational resources. The model's inference speed on CPU devices is also suboptimal, which hinders real-time applications.

Error Propagation in Long Sequences:

The character-level decoding approach accumulates errors over long sequences. For example, misrecognizing a single character in a word (e.g., "কলকাতা” → “কলকাতা”) can distort the entire sequence, leading to a high Sequence Error Rate (SER) of 0.436 compared to the CER of 0.397.

Language Model Dependency:

The system lacks integration with Bangla language models for post-processing, which could correct contextual errors (e.g., “বই” → “বই"). The model produces literal transcriptions without semantic validation, even for nonsensical or misspelled words.

These limitations underscore the need for further refinement to achieve industrial-grade accuracy and scalability. This issue is exacerbated in lengthy documents with dense text.

## 8.3 Future Improvement

Team Up to Grow the Dataset:

Museums, universities, and online communities could help us collect 10,000+ pages of Bangla handwriting from all over Bangladesh—think regional styles like Sylheti or Chittagonian, old manuscripts, and even doctor's notes or legal forms. We could use digital pens to capture handwriting in real time to reduce typos. More data = fewer AI blunders!

Teach the AI to “Clean Up” Messy Pages:

Before transcribing, we will give the AI some smart helpers, such as:

•Scratch Detector: A tool that spots crossed-out text (like scribbles in a diary) and ignores it.

•Layout Detective: An algorithm to map out columns, margins, or notes—so it doesn’t freak out over arrows or side comments.

•Photo Fixer: Fix blurry scans or faded ink using AI.

Add Numbers, Symbols, and English Words:

•Hybrid Texts: “COVID-19” in Bangla? etc

•Loanwords: Automatically convert “টেলিগ্রাম” to “Telegram” for clarity.

Make a version for Phone:

Right now, it is only usable on the desktop. So, we are planning to make a phone version of it also, which will

•Trim the Fat: Remove unnecessary code (like deleting unused apps).

•Simplify Math: Use smaller numbers to save memory.

•Mini-Me Model: Train a tiny AI using tips from the big one.

Add a Bangla “Spell-Check”:

Use tools like BanglaBERT to fix errors humans would catch—like swapping “সকাল” (morning) for “শকাল" (nonsense). Give the AI a grammar book!

# References

**[1]** M. M. R. Nayan and M. A. Haque, "Bangla PDF Speaker: A Complete Computer Application to Convert Bangla PDF to Speech," *2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI)*, Rajshahi, Bangladesh, 2021, pp. 1-5. doi: 10.1109/ACMI53878.2021.9528221.

**[2]** Md. Bhuiyan, M. S. Efaz, T. Reza, A. Ria, Md. T. Reza, and M. I. Hossain, "Segmentation of Bangla Compound Characters: Underlying Simple Character Detection from Handwritten Compound Characters Using YOLOv8," *2024 International Conference on Electrical, Electronics, and Information and Communication Technology (ICEEICT)*, 2024, pp. 1-6. doi: 10.1109/ICEEICT62016.2024.10534334.

**[3]** A. Pal, M. S. Hasan, and S. M. Masudul Ahsan, "Improving Character Recognition in Bangla Handwritten Words: A Two-Stage Single Shot Detector Approach," *2024 International Conference on Advances in Computing, Communication, Electrical, and Smart Systems (iCACCESS)*, Dhaka, Bangladesh, 2024, pp. 1-6. doi: 10.1109/iCACCESS61735.2024.10499463.

**[4]** M. A. N. Mojumder, A. Islam, T. Islam, S. Das, Z. Chowdhury, and A. Rahman, "Bangla and Nagri Language Detection and Classification Using Deep Learning Methods on Text Document Image," *2023 International Conference on Advances in Electronics, Communication, Computing and Intelligent Information Systems (ICAECIS)*, Bangalore, India, 2023, pp. 480-485. doi: 10.1109/ICAECIS58353.2023.10170588.

**[5]** P. Haque, U. Salma, and R. Chowdhury, "Bangla Handwritten Character and Words Recognition-based on the YOLOv5 Algorithm," *2022 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE)*, Naya Raipur, India, 2022, pp. 24-29. doi: 10.1109/WIECON-ECE57977.2022.10151039.

**[6]** A. Ghosh, "Towards Full-page Offline Bangla Handwritten Text Recognition using Image-to-Sequence Architecture," *2023 IEEE Silchar Subsection Conference (SILCON)*, Silchar, India, 2023, pp. 1-6. doi: 10.1109/SILCON59133.2023.10404241.

**[7]** M. F. Mridha, A. Q. Ohi, M. A. Ali, M. I. Emon, and M. M. Kabir, “BanglaWriting: A multi-purpose offline Bangla handwriting dataset,” *Data in Brief*, vol. 34, p. 106633, 2021. doi: [10.1016/j.dib.2020.106633](https://doi.org/10.1016/j.dib.2020.106633).