TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 24, No. 3, June 2026, pp. 945∼956
ISSN: 1693-6930, DOI: 10.12928/TELKOMNIKA.v24i3.26961

A segmentation based optical character recognition system for Bangla printed text

Mahir Mahbub[1,2], Ahmedul Kabir[2]
[1]Department of IoT and Robotics Engineering, Bangabandhu Sheikh Mujibur Rahman Digital University, Gazipur, Bangladesh
[2]Institute of Information Technology, University of Dhaka, Dhaka, Bangladesh

Article Info

Article history:

Received Feb 5, 2025
Revised Dec 8, 2025
Accepted Mar 29, 2026

Keywords:

Masked word prediction
Optical character recognition
Pattern recognition
Spell checker
Textual image segmentation

ABSTRACT

Bangla ranks as the fifth most spoken language globally, catalyzing significant interest in the development of Bangla optical character recognition (OCR) systems. The intricate structure of the Bangla script, including compound characters, modifiers, and headlines, complicates the formation of words. This research introduces a complete OCR system pipeline for printed Bangla text. It employs a thinning-based segmentation approach combined with a convolutional neural network (CNN) to recognize Bangla fonts. Additionally, a part of speech (POS)-aware spell checker is proposed that automatically corrects misspelled words while considering their context within the sentence. We introduce semi-generalized filters that adapt to new fonts, addressing conjunct formation challenges in Bangla OCR. This flexible design allows for adaptation to new fonts. The ResNet50 model is utilized to accurately recognize segmented characters and modifiers. We achieve a character segmentation error of 3.354% and an overall segmentation error of 2.332%. The ResNet50 recognition model achieves an accuracy of 98.345%.

Corresponding Author:

Ahmedul Kabir
Institute of Information Technology, University of Dhaka
Suhrawardi Udyan Rd, Dhaka 1200, Bangladesh
Email: kabir@iit.du.ac.bd

1. INTRODUCTION

Optical character recognition (OCR) converts text images into machine-readable form and is now a core component in digitizing books, archives, administrative records, and everyday documents. Robust OCR systems enable information retrieval and natural language processing and are crucial to preserving linguistic heritage and supporting digital inclusion. Bangla is one of the most widely spoken languages in the world, used by hundreds of millions of speakers, is used as the official or national language in Bangladesh, and is a major language in parts of India. Despite its global significance, Bangla still lacks OCR systems as robust as those available for many languages (e.g., English, German, and Spanish) and even for several Indo-Iranian languages, such as Hindi [1] and Assamese [2]. Existing Bangla OCR often struggles with diverse fonts, noisy documents, and complex page layouts.

Linguistically, Bangla is classified within the Indo-Iranian subgroup of the Indo-European language family and shares lexical and structural features with Hindi and Assamese. As a result, OCR studies on these related scripts offer an important basis. However, Bangla presents unique script-specific difficulties that limit the straightforward adaptation of existing OCR approaches. Bangla uses multiple modifiers (e.g., *Ya-phalā*,

Ra-phalā, *Hras-i-kar*, and *Rēpha*) that vary in position and shape, and its characters and modifiers frequently merge or connect through a common headline (aka *matra*), making character boundaries difficult to localize. These script properties lead to non-trivial segmentation and recognition problems. For example, a single Bangla word can span three vertical zones: an upper zone above the headline, a middle zone containing the primary body of characters, and a lower zone below the baseline, as shown in Figure 1. The headline typically appears as the bar with maximum height in the horizontal projection profile, residing in the upper half of the text line, while the baseline is obtained by scanning the lower portion of the line-segmented image. Complex ligatures and diverse character patterns across fonts further complicate both line/word/character segmentation and accurate classification. Although several Bangla OCR systems have been proposed, persistent performance and compatibility issues especially with complex layouts, varying typefaces, and noisy scans highlight the need for more comprehensive solutions. In this work, we identify key gaps: (i) segmentation methods that are not adaptive and struggle with diverse script patterns during recognition. Limited extensibility for newly observed character formations and emerging typefaces/fonts and (ii) the absence of a contextual and automated progressive spell checker/corrector to refine the raw OCR output.

Our approach addresses these gaps by redesigning the segmentation and recognition pipeline with extensible, filter-based components, ResNet50 recognition and by incorporating an automated spell correction module to improve end-to-end Bangla OCR accuracy.

Figure 1. Illustration of sections for line segmented image

Headline is the connectors of characters in a word, baseline separates the characters and the modifiers below the characters, the upper slice separated by headline is called the ”upper portion”, the part between headline and baseline is called the ”mid portion”, and the part below the baseline is called the lower portion.

Segmentation-based OCR is widely recognized for its flexibility and control. We aim to improve segmentation of printed/typewritten text to reduce linguistic complexity. Research in Bangla OCR is limited, often targeting specific segmentation issues, leading to integration problems and sometimes irreparable errors. Despite these issues, this approach allows customization and control, potentially more effective than scanning/sliding window methods [3], [4]. Authors [5]-[7] pioneered an initial methodology for Bangla OCR as a result of successive research efforts, using eight stroke-based techniques, a fill-circle technique, and a piecewise linear scan for text segmentation. Their method struggles with characters having parts in the upper portion of the headline. For example, Their approach can not segment cases like ◌্র *(Rēpha)* with (ট*(Ti)*, ট*(Tii)*, ি*(Hras-i-kar)*,and ী*(Dirgha-i-kar)*). Their methodology is limited in its ability to separate numerous combined characters such as ট*(Ti)*, and ট*(Tii)*. In our proposed approach, we solve these issues.

Sattar *et al.* [8] suggest segmentation of various characters and modifiers like ি, ী, ট*(Ti)*, ট*(Tii)*, ৈ*(Oi-kar)* ো*(Ou-kar)*, ঠ*(Thi)*, ঠ*(Thii)*, and ◌ঁ*(Chandra-bindu)*, based on distinctive identification. Their approach lacks explicit solutions for the cases merged in the upper portion of headline. Similarly, Akter *et al.* [9] use a scanning-based method with constraints. It makes both approaches font-constrained and inflexible for other Bangla fonts. Our approach is more flexible, supporting filter extensions for easy adaptation to new fonts. Rownak *et al.* [10] propose a curved scanning method for character segmentation to maintain character structure. They use word headline and line headline zones as the two form of headline zones, which help separate numerical characters. However, they sometimes fail to segment ি and ী, especially when merged with others in the upper portion of the headline. Thus, this method adds noise and is limited to static fonts. Zahan *et al.* [11] introduce a two-zone strategy using connected component analysis and a chain approximation algorithm with eight connectivity to separate overlapping characters. However, it also struggles with character overlap and merging issues with the headline. Maung *et al.* [12] present an OCR pipeline for handwritten Bangla that

addresses script-specific challenges, including modifiers, compound characters, and diacritics. Their approach combines you only look once (YOLO) for character detection with EfficientNet-B4 for character recognition, achieving accuracies of 93.87% for grapheme roots and exceeding 98% for both vowel and consonant diacritics. A subsequent Word2Vec-based spelling correction module further reduces the character error rate (CER), enhancing overall system performance. Some researchers used approaches based on support vector machine (SVM), quadratic classifier, and advanced neural network, which have been shown to be promising for their automatic feature extraction capability for OCR [13]-[16].

Spell checkers can be a promising solution to correct the output of the OCR system. UzZaman and Khan [17] utilize double maphony encoding to improve the accuracy of the suggested misspelled words in Bangla. Here, a mapping rule is used that combines the edit distance with a double metaphone. Islam *et al.* [18] proposed method uses the dictionary search, preceded by the suggestion of words based on the edit distance algorithm. They calculate the accuracy using the fractional accuracy method. In their other work, Islam *et al.* [19] integrate the usage of the n-gram language model with the edit distance. Bijoy *et al.* [20] proposed a new detector-purificator-corrector framework, detector–purificator–corrector spell (DPCSpell), based on denoising transformers, and present a method for the creation of large-scale corpus. Erroneous characters are detected and masked by the detector network, and the masked output is then further purified by the purificator. The corrector module is responsible for creating corrections. A new bidirectional encoder representations from transformers (BERT)-based spell checker, progressive stacking, has been developed to improve training and corrective processes for languages such as Bangla [21]. The technique, which uses a convolutional neural network (CNN) sub-model, could reduce training time. They tested on a 6,300-word dataset with spelling errors. Therefore there is no such auto-progressive spell checker for bangla.

We propose several improvements to solve the existing issues while ensuring the future scope for the extension of newly found patterns in characters and new typefaces/fonts. Our proposed segmentation strategy is easy to follow and shows several improvements to previous segmentation issues. We are presenting an OCR system that does not have any compatibility issues related to segmentation and is open for extension to accommodate new fonts. Then we feed the segmented characters to the CNN model to recognize them while preserving the locality of the text position in our textual output. Finally, we feed the output to the proposed spelling checker model, which automatically corrects spelling. This proposed bangla spelling checker is contextual, first auto-progressive, and less sensitive to previous spelling errors.

2. METHOD

This section provides an extensive elucidation of the methodology employed in our research, with comprehensive details presented in the subsequent subsections.

2.1. Preprocessing

OCR preprocessing encompasses several key steps: image binarization, skew correction, noise reduction, and orientation detection. We employed otsu's method for binarization. For skew correction, the approach outlined in [22] was used. Document noise, such as salt and background interference that commonly arises from the age or quality of the document, was mitigated by median filtering. In addition, contrast-limited adaptive histogram equalization (CLAHE) was implemented to enhance local image details, and intensity levels were adjusted using sigmoid correction.

2.2. Segmentation

Here, we define the methodology used to segment the lines, words, and characters within the text images. The subsequent subsections provide a comprehensive explanation of these processes.

2.2.1. Line detection and segmentation

To determine the orientation of an image, we implement both horizontal and vertical scanning techniques to tally the black pixels present for both rows and columns. This pixel count facilitates the generation of a histogram, which is instrumental in line detection. Figure 2 illustrates an example of such a histogram derived from the scanning process. Horizontal scanning of the images is employed to identify lines. Following the separation of these lines, a thinning algorithm is employed which is represented in Figure 3.

Figure 2. The histogram in the vertical and horizontal orientations shows the count of black pixels for each direction respectively. The dotted lines indicate the separation between the lines of text

Figure 3. The separated text line is thinned using skeleton-based thinning approach

2.2.2. Word segmentation

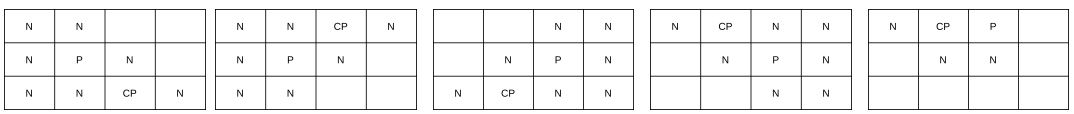
In several word segmentation techniques [5], [6], a word is identified as a single connected component, using vertical histogram scanning to separate words. Although some words, such as খগ*(Khaga)*, lack internal connection to the headline. To address this, we manually extended the horizontal border length of the connected component by 10% to its vertical length. We did it because our results indicate that the intercharacter spacing is usually 6-10%, while the interword spacing is 14-25% of the word's vertical length in standard text. Then filled black rectangles are superimposed on the modified border of the connected components in a copy of the image. Then we recalculate the border of the connected component. Finally, we extract the areas from the main image using recalculated borders, which result in images of words. The outcome of the method can be seen in Figure 4. Two separate components (আ, and গ) showed in Figure 4(a), of the word আগে*(Āgē)* are merged into a single component using the filled rectangle in Figure 4(b).

(a) (b)

Figure 4. An example of word segmentation by the filled rectangle: (a) connected component: the two words are segmented into three connected components. Those components are shown inside the rectangles and (b) extended and filled the horizontal length of the rectangle covering the word is extended and filled. Therefore the sub-section of the words are merged and separated as a single word

2.2.3. Character segmentation

We introduce common search patterns through what we call filters to segment the characters and modifiers. We designed the filters in such a way that the filters do not overlap with each other when searching. The filters in Figure 5 remove the headline from the thinned image. Here, CP denotes the start of the pixel from which we start searching. P and N denote positive and negative character pixel. The pixel colors for CP and P are black. Patterns matching the filters in Figures 5(a) to (e) indicate that there is only a headline. Thus, we remove the black pixels of CP and P. Filters are crucial because the headline after thinning is not always a straight line, as can be seen in Figure 6. The small rectangles in Figure 6 reveal those variations in the structure of the headline. Thus these filters help identify the headline so that they can be removed.



(a) (b) (c) (d) (e)

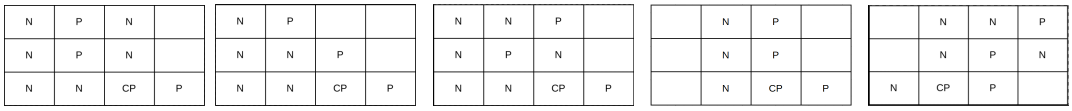
Figure 5. Filters for headline removal: (a) filter 1, (b) filter 2, (c) filter 3, (d) filter 4, and (e) filter 5

Figure 6. Variations in the headline appear following the thinning process, as indicated within the gray rectangles

Our approach effectively resolves merging issues, such as touching and intersecting, as shown in Figure 7. We use filters to identify and separate characters and modifiers like ◌্র *(Rēpha)*, ◌ি, ◌ী, ◌ঁ, and upper portions of ট, ই, ঈ, উ, and ঊ as shown in Figure 7(a) and 7(b). In Figures 8(a) to (e), the filters in Figures 8(a) to (c) are used to search ◌ি in the image. Upon identifying these search patterns with the filters, we search for single "up", "up-right", and "right" pixels from the current pixel accordingly. At first we search in the straight "up" direction. If the expected pixel is not found, the search moves to the up-right and then right. We do this single pixel search because sometimes the pixels of the thinned headline reside one pixel "up" from the average headline. It usually happens when characters join with the headline. So, the starting point is adjusted to the new positive search point upon matching this search pattern.

(a) (b)

Figure 7. Examples of character merging issues: (a) ◌্র with upper portion of ট and (b) ◌ি with upper portion of ট



(a) (b) (c) (d) (e)

Figure 8. Filter for detecting upper modifiers using different search and erasing strategies: (a) filter 1, (b) filter 2, (c) filter 3, (d) filter 4, and (e) filter 5

Then we search using depth-first search (DFS) from the identified headline pixel. When a horizontal pattern or an intersecting point is found with at least a six-pixel count during the DFS search, we draw a white filled rectangle that separates the merging characters (◌ি and ◌্র *(Rpha)*), as shown in Figure 7(a). A ◌্র*(Rēpha)* possibly merges with a headline when matched to the filter in Figure 6 which we previously used for adjustment of headline pixel. We replace CP and P with N, then apply DFS again using the "up" and "up-left" patterns. The intersection point is discovered by finding at least three positive adjacent joint pixels in the upper portion of the headline. It separates ◌্র*(Rēpha)*. Likely, ◌ী, ◌ঁ and upper portion of ট, ই, ঈ, উ, ঊ exist here when the matched pattern with the filter in Figure 8(d). Preceeded by a pattern corresponding with any of the Figures 8(a)-8(c). Therefore, we draw a white line, which is 20% of the width (aka. vertical length) of the line-segmented picture, just one pixel up from the CP pixel at the backward horizontal direction. This white line separates ◌ী from the headline without any deformation.

Again, similar to the previous one, we conduct a DFS based on an "up", "up-left", "left" pattern. We remove the final positive search pixel after locating the intersection point. It separates characters/modifiers to prevent merging. The whole process is described in Algorithm 1. Our method splits ট, ই, ঈ, উ, ঊ into two parts, as shown in Figure 9. Segmented characters are collected on the basis of their centroids. From the centroid, the borders of the characters are calculated.

Figure 9. An example of fully segmented text. The images are separated into lines. Then the lines are separated into words. The words are separated into characters and modifiers

Algorithm 1 Separate characters and modifiers from OCR image

procedure SEPARATINGCHARACTERSANDMODIFIRES($imageArray, lineRowIndex$)
  lineArray = &$ImageArray[LineRowArray]$
  lineLength = $length(lineArray)$
  for $i = 0$ to $lineLength - 1$ **do**
    filterType, horizontalCount, erasingIndex = $FindSeperatorFilter(CP = \&LineArray[i])$
    if $filterType$ in $[a, b, c]$ **then**
      if $horizontalCount \geq 6$ **then**
        $EraseByRectangle(rectangleStartIndex = rectangleIndex)$
      end if
      innerFilterType, _, erasingIndex = $FindSeperatorFilter(CP = \&LineArray[i])$
      if $innerFilterType$ == d **then**
        $EraseByBackwardLine(index = erasingIndex)$
        isMergeFound, mergerPixel = $SearchForMerge(CP = \&LineArray[i])$
        if $isMergeFound$ **then**
          $ErasingMergerPixels(mergerPixel = mergerPixel)$
        end if
      end if
    else if $filterType$ == e **then**
      PIndexes = $GetPIndexes(CP = \&LineArray[i])$
      isMergeFound, mergerPixel = $SearchForMerge(CP = \&LineArray[i])$
      if $isMergeFound$ **then**
        $ErasingMergerPixels(mergerPixel = mergerPixel)$
      end if
      $ErasingPositivePixels(CP = i, PIndexes = PIndexes)$
    end if
  end for
end procedure

2.3. Recognition and post processing

Our proposed model is a standard ResNet50 CNN (pre-trained with imagenet) and two additional feed-forward layers for character recognition. The model is trained with 127 classes with 10 fold cross validation. Each class is constrained to 451 labeled instances of images because of the rare occurrence of some characters/modifiers. We have experimented using other CNN models. Every class has We merge character output into words and lines. Post-processing corrects certain OCR generated errors. We handle the joining character ◌ৎ separately by merging it with the previous character, bypassing the recognizer. After headline erasing, ত (the letter *Ta*) and ৩ (the number 3) both become ৩. Therefore we check for several cases to determine if it is ত or ৩. If the ith value in word is ৩, the cases are as follows: If the $(i-1)$th or $(i+1)$th value is a number, the ith value is ৩. If the $(i+1)$th value is য় and the $(i-1)$th value is not a number or the $(i-1)$th value does not exist, the ith value is ৩. Otherwise, the ith value is ত. Only unique cases in ত ends with ম. In addition, erasing the headline resulted in the segmentation of ই, উ, ট, ঈ, ঊ into two parts. Among the segmentation of ই, উ, ট can result in similarity to other characters. 'ই' becomes 'হ' and 'ে', 'উ' becomes 'ড' and 'ে', 'ট' becomes 'ঢ' and 'ে'. So, when these characters are found in the ith position, we check for 'ে' in the $i-1$th position and take the decision accordingly.

2.4. Spell checker

The spell checker we propose processes text obtained via OCR and outputs text with corrected spelling. The architecture of the spell checker is detailed in Figure 10 and is further elaborated upon in subsequent sections. Our approach utilizes a BERT, which employs masked word prediction to infer words by considering the surrounding context. This mechanism generates a list of likely words to correct spelling errors. The BERT model is a base-uncased model, featuring 12 layers, 768 hidden units, 12 attention heads, and a total of 110 million parameters.

The primary training corpus for the BERT model is sourced from Bangla newspaper articles, including Daily Prothom Alo,Daily Kalerkontho, Daily Ittefaq and Bangla Wikipedia, constituting 65.36% of the dataset, with the remaining 34.64% derived from *Bangla Wikipedia*. Data acquisition was performed through web crawling and manual collection between 2018 and 2020, resulting in a corpus comprising 10,340,273 sentences and 8,928,472 unique words. In Table 1 the data source is described using its proportional value from different sources.

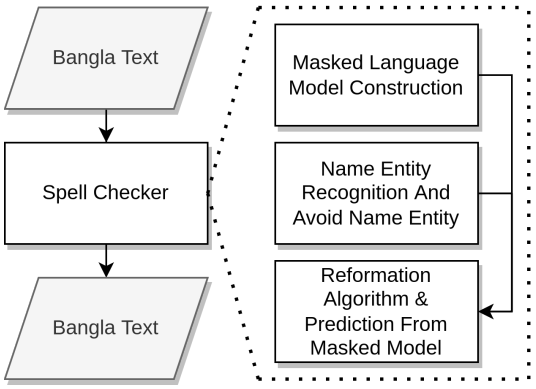


Figure 10. The main components of the proposed spell checker are shown inside the dotted area. The spell checker takes the sentence as input and produces the sentence with the corrected spelling of words

Table 1. Details of the dataset used for corpus building

Data source	Bangla newspaper	Bangla Wikipedia
Proportion(%)	65.36	34.64

2.4.1. Error detection

We address spelling errors in the OCR output, first by identifying named entities using an mBERT model (https://huggingface.co/sagorsarker/mbert-bengali-ner). Non-named entities undergo a dictionary lookup within our 451,742-word database. Words not found and not numerical are flagged as potential misspellings for further analysis. To correct the spelling, we took a two-step approach. Firstly, we took the prediction for the probable word list from the masked word prediction model for probably misspelled words. The predicted words then undergo a correction process based on Levenshtein distance (LD). LD calculates the minimum edits needed for the conversion of word $s1$ to word $s2$. Let I represent the current location in the input string $s1$ and J represent the current location in the output string $s2$. If the probable *misspelled word (mw)* length is less than 4, we consider the maximum edit distance (ml) of 2 indicated by ml to select a correctly spelled candidate word. If the mw or predicted word(pw) has complex characters (two characters joined by "্"), the value of ml is extended by 1. For the value of the length of mw $>= 4$, the value of ml is half the length of mw. Here, the selection of minimum edit distance is based on analysis and experiment. When predicted words from the masked model predicted word list fulfill the criterion described above, the word from the list with the minimum edit distance is considered as the correctly spelled word and replaces the probable misspelled word. The model can also suggest the most probable list of correctly spelled words.

3. RESULTS AND DISCUSSION

Here, we discuss the result of our segmentation process. Since our filters are currently tailored to the specific fonts used in the Daily Prothom Alo and Somokal newspapers, it is important to compare how well the method generalizes across different typefaces. The segmentation performance for the different fonts is shown in Table 2. Our proposed approach has segmented printed text images into 123,422 characters/modifiers.

The segmentation performance for the different fonts is shown in Table 2. Our proposed approach has segmented printed text images into 123,422 characters/modifiers. The majority of characters, such as গ, থ which are occasionally aligned above or with the headline, are responsible for the character-level segmentation problem. For the Daily Prothom Alo and Somokal typeface, an additional skew correction of the headline after the thinning procedure resulted in a 2.26% reduction of error. In our proposed approach, we solve many merging-related segmentation issues.

Table 2. Comparative result of character segmentation for different newspaper fonts

Font source	Character seg. err(%)	Total seg. character
Prothom Alo	2.80	63,359
Somokal	4.45	32,112
Jugantor	11.38	27,951

In Table 3, we compare the performance of character-level segmentation across different newspaper typefaces. This comparison is important because it reveals certain widely used newspaper fonts, especially with complex ligatures and closely spaced glyphs. It shows that extension to a new font face is important for future generalization, which scope we opened through the filtering-based approach. In our method, the average segmentation error rate is 0.22% on all typefaces.

Table 3. Result of segmentation of some merging issues and characters/modifiers. Most of the issues arise due to the merging of modifiers and characters

Issues	Character seg. err(%)
Merging with ি(*Hras-i-kar*)	0.95
Merging with ী(*Dirgha-i-kar*)	1.42
ি(*Hras-i-kar*)	0.21
ী(*Dirgha-i-kar*)	0.13
ৈ(*Oi-kar*)	0.02
ৌ(*Ou-kar*)	0.23
ঁ(*Chandra-bindu*)	0.48
Merging with র্(*Rēpha*)	0.03
্য(*Ya-phalā*)	0.41
্র(*Ra-phalā*)	6.31
গ(*ga*)	8.09
থ(*tha*)	7.23

Table 4 provides a direct quantitative comparison between our proposed method and existing Bangla OCR systems under the same experimental conditions and font settings (Prothom Alo and Somokal). This comparison is crucial to demonstrate that performance gains in character segmentation and recognition are not anecdotal, but are consistently achieved despite the limited and highly imbalanced 127-class data set. Each class is constrained to 451 labeled instances of images because of the rare occurrence of some characters/modifiers. Our approach has an error rate of only 3.35%, which is significantly lower than that of the previous methods.Given the scarcity of complete OCR approaches in the literature, the final results are compared exclusively with studies that employ full OCR methods. In addition, there is a limited standard benchmark for comparing the performance of OCR [23], [10].

We have experimented with different CNN models for recognizing characters/modifiers, as shown in Table 5. The ResNet50 models gives the best outcome compared to other evaluated CNN models with a accuracy of 98.345% and an F1 score of 96.854%. We evaluate the performance of the spell checker on the Bangla BERT masked word model described above. The metrics used to evaluate the models are described:

- TP (true positive): did not change the correct word/total correct word
- FN (false negative): change the correct word incorrectly/total correct word
- FP (false positive): did not change the incorrect word (Mark incorrect as correct)/total incorrect word.
- TN (true negative): change the incorrect word correctly/total incorrect word
- TN_PLUS: change the incorrect word incorrectly/total incorrect word

Table 4. Comparative analysis of the traditional segmentation-based Bangla OCR systems

Literature	Character seg. error (%)	Final outcome error (%)
Our proposed approach	3.35	2.33
Chowdhury and Pal [7]	39.87	44.53
Mahmud *et al.* [24]	34.22	-
Sattar *et al.* [8]	36.99	-
Akter *et al.* [9]	42.66	-
Omee *et al.* [23]	28.60	30.65
Rownak *et al.* [10]	23.54	-
Zahan *et al.* [11]	28.99	-

Table 5. The result of the different CNN model on labeled segmented bangla characters/modifiers

Model name	Accuracy (%)	F1
ResNet50	98.345	96.854
VGG-16	86.433	84.121
Inception V2	89.992	85.471
ANN(10 Convolutional Layers)	67.443	59.328

The dataset utilized for the evaluation of spell checker performance is constructed in accordance with the methodology described in the literature by Sifat *et al.* [25]. There are 10244 misspelled words and 52334 correctly spelled words in the dataset. We expect to have the value of TP, and TN as large as possible and others as low as possible. We experimented with various *maximum edit distance values (ml)*. We defined and checked some conditions to decide when we should change the spelling of words based on the LD. The conditions are listed:

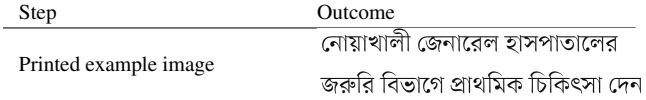
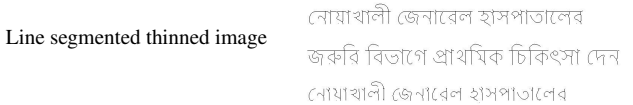
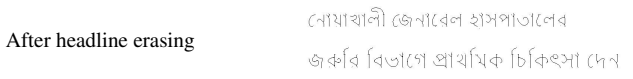
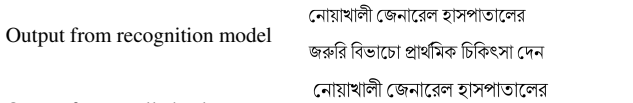
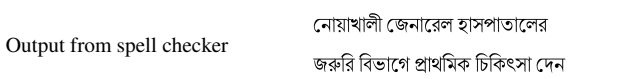
- C1: ml = (probable misspell word aka mw's length(length(mw)))/2
- C2: ml = length(mw)/2 if length(mw) $>$ 4 else ml = 2
- C3: ml = length(mw)/2 if length(mw) $>$ 6 else ml = 2
- C4: ml = length(mw)/2 if length(mw) $>$ 6 else ml = 3

In Table 6, C2 shows better results than other conditions that have already been mentioned in the above proposed method. Thus we select C2 as our parameterized value for ml. Although C2 shows 8.33% in TP_PLUS, the value of TN is 50.17% which is significantly good. C1 performs better in detecting the correct words in sentences. Finally, in Table 7, we show an example to demonstrate the step-by-step process of our OCR. It shows the different outputs of the OCR through its work cycle.

Table 6. The outcomes of our spell checker using Bangla BERT under various conditions

Condition	TP	FN	FP	TN	TN_PLUS
C1	0.983	0.016	0.677	0.320	0.001
C2	0.978	0.021	0.415	0.501	0.083
C3	0.977	0.022	0.553	0.441	0.005
C4	0.962	0.037	0.649	0.201	0.149

Table 7. Detailed view of the pipeline's processes using an example

Step	Outcome
Printed example image	
Line segmented thinned image	
After headline erasing	
Output from recognition model	
Output from spell checker	

So among the tested fonts, the proposed method achieved the lowest segmentation error with Prothom Alo (2.80%), while fonts like Jugantor saw higher error rates due to the filters are not optimized for them. Compared to previous systems with error rates, our approach reduced character-level errors to 3.35% and final text errors to 2.33%. The part of speech (POS) sentitive autoprogressive and contextual BERT-based spell checker further improved recognition outcomes, particularly under condition C2.

4. CONCLUSION

In this work, we have addressed the problem of digitizing Bangla printed images through a complete OCR pipeline, with particular emphasis on character segmentation, recognition, and post-processing. We proposed a modular, extendable filter-based segmentation method that mitigates many of the shortcomings found in existing segmentation-driven Bangla OCR approaches, thus reducing segmentation-related errors across various printed fonts. By integrating our segmentation output with ResNet50, we achieved competitive character-level recognition performance, despite the challenges posed by the complex Bangla script and compound characters. We proposed a contextual spell checker built on a masked language model (BERT). By operating at the phrase level, this component enhances the robustness of the OCR pipeline against both recognition errors and traditional spelling errors, and showcases the utility of modern language models in low-resource script OCR.

Although our proposed complete pipeline does not solve all challenges of Bangla OCR, it establishes a strong foundation and suggests several promising directions. Our current system targets printed text. Extending the segmentation, recognition, and contextual correction components to handle handwritten Bangla, where the variability is significantly higher, remains a significant and impactful research direction. The performance of both recognition and BERT-based spell correction can be substantially improved by training and evaluating larger, more diverse datasets that better cover rare compound characters, multiple fonts, and a wider range of domains. Developing additional filters and integrating them into the existing systems can enhance the overall coverage and robustness of the pipeline. Also, there are a large number of compound characters in Bangla. These characters appear too rarely to be incorporated into our recognition model. These characters affect the overall performance of the OCR. Although our primary objective was to enhance segmentation techniques, it is essential to acknowledge that several contemporary studies employing non-segmentation methodologies have garnered significant recognition and demonstrated high efficacy.

In summary, our work presents a segmentation-centric pipeline for Bangla printed OCR, combining classical image processing and modern deep learning with contextual language modeling. We expect that the methods and insights introduced here will facilitate further research in robust Bangla OCR and, more broadly, OCR for other complex Indic scripts.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENTS

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Mahir Mahbub	✓	✓		✓	✓	✓		✓	✓	✓	✓			
Ahmedul Kabir	✓	✓			✓		✓		✓	✓		✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal Analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject Administration
Fu : **Fu**nding Acquisition

CONFLICT OF INTEREST

Authors state no conflict of interest.

DATA AVAILABILITY

Data supporting the findings of this study are available upon request from the corresponding author.

REFERENCES

[1] P. P. Bairagi, "Optical Character Recognition for Hindi," *International Research Journal of Engineering and Technology*, vol. 5, no. 5, pp. 3968–3973, 2018.

[2] N. Borpuzari and A. K. Mahanta, "A Framework for Pre Processing, Recognizing and Distributed Proofreading of Assamese Printed Text," in *2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)*, IEEE, Sep. 2021, pp. 1–7, doi: 10.1109/ICSES52305.2021.9633818.

[3] D. Paul and B. B. Chaudhuri, "A BLSTM Network for Printed Bengali OCR System with High Accuracy," *arXiv*, Aug. 2019, doi: 10.48550/arXiv.1908.08674.

[4] W. Shang, L. Chen, and W. Xiang, "Yi printed character recognition based on deep learning," *Procedia Computer Science*, vol. 242, pp. 584–591, 2024, doi: 10.1016/j.procs.2024.08.111.

[5] U. Pal and B. B. Chaudhuri, "OCR in Bangla: An Indo-Bangladeshi language," in *Proceedings - International Conference on Pattern Recognition, IEEE Comput. Soc. Press*, 1994, pp. 269–273, doi: 10.1109/icpr.1994.576917.

[6] B. B. Chaudhuri and U. Pal, "OCR system to read two Indian language scripts: Bangla and Devnagari (Hindi)," in *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, IEEE Comput. Soc,* 1997, pp. 1011–1015, doi: 10.1109/icdar.1997.620662.

[7] B. B. Chaudhuri and U. Pal, "A complete printed Bangla OCR system," *Pattern Recognition*, vol. 31, no. 5, pp. 531–549, Mar. 1998, doi: 10.1016/S0031-3203(97)00078-2.

[8] M. A. Sattar, K. Mahmud, H. Arafat, and A. F. M. Noor Uz Zaman, "Segmenting Bangla text for optical recognition," in *2007 10th International Conference on Computer and Information Technology, ICCIT, IEEE*, Dec. 2007, pp. 1–6, doi: 10.1109/ICCITECHN.2007.4579373.

[9] N. Akter, S. Hossain, M. T. Islam, and H. Sarwar, "An algorithm for segmenting modifiers from bangla text," in *Proceedings of 11th International Conference on Computer and Information Technology, ICCIT 2008, IEEE*, Dec. 2008, pp. 177–182, doi: 10.1109/ICCITECHN.2008.4803049.

[10] A. F. Rownak, M. F. Rabby, S. Ismail, and M. S. Islam, "An efficient way for segmentation of Bangla characters in printed document using curved scanning," in *2016 5th International Conference on Informatics, Electronics and Vision, ICIEV 2016, IEEE*, May 2016, pp. 938–943, doi: 10.1109/ICIEV.2016.7760138.

[11] T. Zahan, M. Z. Iqbal, M. R. Selim, and M. S. Rahman, "Connected Component Analysis Based Two Zone Approach for Bangla Character Segmentation," in *2018 International Conference on Bangla Speech and Language Processing, ICBSLP 2018, IEEE*, Sep. 2018, pp. 1–4, doi: 10.1109/ICBSLP.2018.8554684.

[12] A. T. Maung, S. Salekin, and M. A. Haque, "A hybrid approach to Bangla handwritten OCR: combining YOLO and an advanced CNN," *Discover Artificial Intelligence*, vol. 5, no. 1, p. 119, Jun. 2025, doi: 10.1007/s44163-025-00251-7.

[13] M. G. Mortuza, S. Islam, M. H. Kabir, and U. Chong, "A Convolutional Neural Network-based Approach to Recognize Bangla Handwritten Characters," in *Computer Vision and Image Analysis for Industry 4.0,* 2023, pp. 150–163, doi: 10.1201/9781003256106-13.

[14] M. Raquib, M. A. Hossain, M. K. Islam, and M. S. Miah, "VashaNet: An automated system for recognizing handwritten Bangla basic characters using deep convolutional neural network," *Machine Learning with Applications*, vol. 17, p. 100568, Sep. 2024, doi: 10.1016/j.mlwa.2024.100568.

[15] A. Hazra, P. Choudhary, S. Inunganbi, and M. Adhikari, "Bangla-Meitei Mayek scripts handwritten character recognition using Convolutional Neural Network," *Applied Intelligence*, vol. 51, no. 4, pp. 2291–2311, Apr. 2021, doi: 10.1007/s10489-020-01901-2.

[16] M. Aktar, N. Islam, and C. Yang, "Bangla Character Detection Using Enhanced YOLOv11 Models: A Deep Learning Approach," *Applied Sciences (Switzerland)*, vol. 15, no. 11, p. 6326, Jun. 2025, doi: 10.3390/app15116326.

[17] N. UzZaman and M. Khan, "A Double Metaphone encoding for Bangla and its application in spelling checker," in *Proceedings of 2005 IEEE International Conference on Natural Language Processing and Knowledge Engineering, IEEE NLP-KE'05, IEEE*, 2005, pp. 705–710, doi: 10.1109/NLPKE.2005.1598827.

[18] M. I. K. Islam, R. I. Meem, F. B. A. Kasem, A. Rakshit, and M. T. Habib, "Bangla Spell Checking and Correction Using Edit Distance," in *1st International Conference on Advances in Science, Engineering and Robotics Technology 2019, ICASERT 2019*, IEEE, May 2019, pp. 1–4, doi: 10.1109/ICASERT.2019.8934536.

[19] M. I. K. Islam, M. T. Habib, M. S. Rahman, M. R. Rahman, and F. Ahmed, "A context-sensitive approach to find optimum language model for automatic Bangla spelling correction," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 11, pp. 184–191, 2018, doi: 10.14569/ijacsa.2018.091126.

[20] M. H. Bijoy, N. Hossain, S. Islam, and S. Shatabda, "A transformer-based spelling error correction framework for Bangla and resource scarce Indic languages," *Computer Speech and Language*, vol. 89, p. 101703, Jan. 2025, doi: 10.1016/j.csl.2024.101703.

[21] D. D. Banik, S. Das, S. Martha, and A. Shankar, "BERT-Inspired Progressive Stacking to Enhance Spelling Correction in Bengali Text," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 23, no. 8, pp. 1–12, Aug. 2024, doi: 10.1145/3669941.

[22] B. B. Chaudhuri and U. Pal, "Skew angle detection of digitized indian script documents," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 182–186, 1997, doi: 10.1109/34.574803.

[23] F. Y. Omee, S. S. Himel, and A. N. Bikas, "A Complete Workflow for Development of Bangla OCR," *International Journal of Computer Applications*, vol. 21, no. 9, pp. 1–6, May 2011, doi: 10.5120/2543-3483.

[24] S. M. M. Mahmud, N. Shahrier, A. S. D. Hossain, M. T. C. Mohmud, and M. A. Sattar, "An Efficient Segmentation Scheme for the Recognition of Printed Bangla characters," in *Proceedings of ICCIT*, 2003, pp. 283–286.

[25] M. H. R. Sifat *et al.*, "Synthetic Error Dataset Generation Mimicking Bengali Writing Pattern," in *2020 IEEE Region 10 Symposium (TENSYMP)*, IEEE, 2020, pp. 1363–1366, doi: 10.1109/TENSYMP50017.2020.9230838.

BIOGRAPHIES OF AUTHORS

Mahir Mahbub received his B.Sc. and M.Sc. degrees in Software Engineering from the Institute of Information Technology (IIT) at the University of Dhaka, Bangladesh. Currently, he holds a lecturer position in the Department of IoT and Robotics Engineering at Bangabandhu Sheikh Mujibur Rahman Digital University, Bangladesh. His research areas include natural language processing (NLP), software engineering, cybersecurity, and machine learning. He can be contacted at email: mahbub0001@bdu.ac.bd.

Ahmedul Kabir is an Associate Professor at the Institute of Information Technology (IIT), University of Dhaka. He earned his Ph.D. in Computer Science from Worcester Polytechnic Institute (WPI) in Massachusetts, USA, focusing on innovative ensemble machine learning algorithms. During his Ph.D., he also collaborated with the University of Massachusetts Medical School on health informatics. His research areas include natural language processing, computer vision, data mining, and software analytics. He has co-authored numerous research papers in both national and international journals and conferences and has guided several M.Phil, Master's, and Bachelor's theses and projects. He can be contacted at email: kabir@iit.du.ac.bd.