Unleashing the Power of YOLOv8 for Accurate Bengali Mathematical Expression Detection

Izazul Hoq Imran (irkfcb@gmail.com)  
East West University

Tropa Das  
East West University

Asadullah Hel Galib  
East West University

Tanni Roy  
East West University

Research Article

Keywords: landscapes, Green Building, Path-finding, Optimization, environment

Posted Date: May 4th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2759436/v3

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License

Additional Declarations:  
No competing interests reported.

Graph 1 is available in the supplementary files section.
Unleashing the Power of YOLOv8 for Accurate Bengali Mathematical Expression Detection

Abstract. The recognition of Bengali handwritten digits is an intriguing and challenging research area, requiring an accurate and efficient approach that can localize and recognize multiple objects. This field has attracted considerable attention from experts in pattern recognition. This paper proposes a novel approach for accurately detecting and localizing Bengali handwritten digits and operators in an image using YOLOv8 object detection algorithm. One of the key advantages of YOLOv8 is that it not only detects objects quickly, but also returns localization data, such as the position of each object within the image, which is crucial for identifying the line where the digit or sign belongs. By analyzing the position of each character, this method is able to convert handwritten mathematical equations into digital text, which is an important task for various applications in the fields of education, science, and engineering. The proposed method eliminates the use of anchor boxes, which can lead to distribution challenges, and achieves promising accuracy on a custom dataset. The paper also introduces a new dataset, “BanglaOngko” designed specifically for Bengali handwritten digits and object detection. The results demonstrate the effectiveness of the YOLOv8 model for Bengali digit and operator recognition and suggest its potential for other similar recognition tasks. The proposed method and dataset could be useful for further research and development in Bengali handwriting recognition.

Keywords: Bangla OCR, Bengali handwritten digit, Object detection, YOLOv8, Mathematical Expressions, Digit Recognition, Object Localization, OCR.

1 Introduction

The advancements in deep learning have greatly improved the field of computer vision, particularly in Optical Character Recognition (OCR), which involves recognizing printed or handwritten text and converting it into machine-readable format [1]. This paper proposes a novel OCR model to detect handwritten Bengali digits and operators using the state-of-the-art object detection algorithm YOLOv8. YOLOv8, or “You Only Look Once” version 8, is an anchor-free model that predicts the center of an object directly, rather than offsetting it from a pre-defined anchor box. By utilizing YOLOv8, we can achieve precise localization of every character within an image and leverage its position to determine the line where the digit or sign belongs. This approach ensures accurate and efficient
detection and recognition of handwritten Bengali characters, representing a significant step forward in the development of advanced OCR technology. In recent years, there has been significant research focused on recognizing handwritten Bengali characters and digits as Bengali is the seventh most spoken language in the world, with over 300 million speakers worldwide [2]. Developing a reliable Bengali handwritten character recognition and localization system is a challenging task, but important for improving OCR technology for better communication and accessibility for Bengali speakers. In this project, we have worked with bengali digits, parentheses, and some mathematical operators (such as +,-,x,/ etc. ). By analyzing the position of each character, the model is able to convert handwritten mathematical equations into digital text, which is an important task for numerous applications in the fields of education, science, and engineering.

Previous research on recognizing handwritten Bengali digits has predominantly focused on object classification without incorporating localization information or considering the presence of multiple objects in a single image. However, this approach has significant limitations that hinder the development of OCR technology for Bengali. To address this issue, we have introduced a new dataset, “BanglaOngko” containing 21 classes of Bengali digits and operators. Each image is annotated with precise bounding box information, enabling object localization. We can use this newly developed dataset to train and test object detection models that more accurately recognize and locate handwritten Bengali digits and operators, ultimately driving progress in the field of OCR technology for Bengali.

2 Background and Related Works

The background and related works section is divided into the following parts.

1. Previous works related to handwritten digit recognition.
2. Previous research on Bengali handwritten digit recognition and mathematical operator recognition.
3. Advantages of our method compared to previous works on Bengali digit recognition and localization.

2.1 Previous Works Related to Handwritten Digit Recognition

Yuli Wu, Yucheng Hu, Suting Miao went with an object detection based approach, using a Cascade R-CNN network for handwriting localization [3]. They trained their dataset with 600 grayscale images that contained both handwritten and printed data. E. Kavallieratou et al. proposed a system for handwritten text localization in document images, which performs skew angle correction and localizes handwritten areas based on measures of regularity in shape and dimensions [4]. The proposed technique was tested on a variety of documents and had a success rate of over 88%. Another model that is customized faster
regional convolutional neural network (Faster-RCNN) is introduced, comprising of three steps: region of interest annotation, improved Faster-RCNN with DenseNet-41, and regression and classification layer for digit localization and classification [5]. The proposed method’s performance was evaluated on the standard MNIST database.

2.2 Previous research on Bengali handwritten digit and mathematical operator recognition.

Earlier research on Bengali handwritten digit recognition relied on conventional machine learning techniques such as k-nearest neighbors (KNN) [6] and support vector machines (SVM) [7]. BengaliNet, a cost-effective convolutional neural network architecture was presented for recognizing Bengali characters [8]. The model had a parameter range of 2.24 to 2.43 million and surpassed the performance of prior works on eight variations of CMATERdb datasets. The proposed architecture also achieved remarkable overall accuracies ranging from 96-99% on other Bengali character datasets such as Ekush, BanglaLekha, and NumtaDB datasets. A novel tri-level segmentation approach was proposed for Optical Character Recognition (OCR) of Indic scripts, with a specific focus on Bangla [9]. The presence of modified and compound characters made this task challenging. However, the researchers were successful in achieving favorable segmentation results when the proposed method was tested on a dataset consisting of 50 handwritten text documents.

2.3 Advantages of our method compared to previous works on Bengali digit recognition and localization

Most prior research focuses on single Bengali handwritten character/digit recognition using various machine learning techniques, such as Support Vector Machines (SVM), K-nearest neighbors (KNN), Artificial Neural Networks (ANN), and Convolutional Neural Networks (CNN), along with morphological and graph-based methods. However, these methods are not capable of detecting multiple digits and their positions from a single image. Our proposed method, on the other hand, can quickly detect multiple digits from an image with their coordinates, eliminating the need for an extra line detection machine learning method. From these coordinates, we can determine the lines where a digit belongs, enabling us to convert the handwritten text to digital text accurately.

3 Methodology

3.1 Data Collection

We captured the images using a mobile-phone camera from various sources such as friends, school students, and family members. We have collected approximately 300 images of 21 classes that include Bengali digits, operators, and parentheses. Table 1 the count of each objects in the dataset where the total object count is 7372.
### Table 1: Object count in Dataset

<table>
<thead>
<tr>
<th>Class</th>
<th>Object Count</th>
<th>Class</th>
<th>Object Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>258</td>
<td>equal</td>
<td>368</td>
</tr>
<tr>
<td>1</td>
<td>487</td>
<td>(</td>
<td>375</td>
</tr>
<tr>
<td>2</td>
<td>434</td>
<td>)</td>
<td>362</td>
</tr>
<tr>
<td>3</td>
<td>258</td>
<td>minus</td>
<td>528</td>
</tr>
<tr>
<td>4</td>
<td>348</td>
<td>multiplication</td>
<td>386</td>
</tr>
<tr>
<td>5</td>
<td>310</td>
<td>plus</td>
<td>427</td>
</tr>
<tr>
<td>6</td>
<td>401</td>
<td>{</td>
<td>308</td>
</tr>
<tr>
<td>7</td>
<td>410</td>
<td>}</td>
<td>275</td>
</tr>
<tr>
<td>8</td>
<td>358</td>
<td>[</td>
<td>309</td>
</tr>
<tr>
<td>9</td>
<td>307</td>
<td>]</td>
<td>309</td>
</tr>
<tr>
<td></td>
<td>division</td>
<td></td>
<td>430</td>
</tr>
</tbody>
</table>

### 3.2 Binarize Image

We converted the color images to grayscale using the BGR to Gray conversion method. Then, we applied Gaussian blur with a kernel size of 3x3 to remove any noise present in the image. Finally, we binarized the processed image to keep the digits in black and the background in white. The YOLO object detection algorithm requires a minimum image size that is a multiple of 32 due to downsampling that occurs during the CNN architecture. Downsampling reduces spatial dimensions of the feature map, enabling detection of objects at different scales. To prevent rounding errors during downsampling, it’s recommended to use image sizes that are multiples of 32 (e.g. 320x320, 416x416, 640x640) when training a YOLO model. Hence, we resized all the images to 640 x 640 pixels to ensure consistency in image sizes for further processing and analysis.

### 3.3 Annotation

As the model aims to detect multiple objects, first, objects should be localized in an image. For this, we used Roboflow [10] to annotate the images manually. We drew a bounding box around each object (digits, operators, and parentheses) and classified them into 21 different classes: 0 to 9, addition, subtraction, multiplication, division, equal, and brackets.
Convolutional neural networks (CNN) have proved to be highly effective in object detection algorithms, with models such as R-CNN, Fast R-CNN, Faster R-CNN, SSD, RetinaNet, and YOLO being developed over the years. Prior to YOLO, these models employed a two-stage detection process, involving selective search or regional proposal network to identify interesting regions, which were then processed by a classifier. However, YOLO was designed to handle detection as a regression problem and performed both detection and classification with a single neural network, making it the fastest one-stage detector available.

The latest version of YOLO is YOLOv8, which prioritizes speed, size, and accuracy. It incorporates new features like a new backbone network, anchor-free split head, and loss functions for improved performance while maintaining a compact size and exceptional speed. YOLOv8 is capable of supporting various vision AI tasks, including detection, segmentation, pose estimation, tracking, and classification. It offers superior performance and accuracy, making it a valuable tool for object detection and image segmentation. Fig. 2 illustrates the architecture of YOLOv8.

3.4 YOLOv8
3.5 Training YOLOv8

The training settings for YOLO models refer to the various hyperparameters and configurations used during the training process to optimize the model’s performance on a given dataset. These settings are typically tuned through careful experimentation and analysis to achieve the best possible results. Table 2 displays the training results mAP (mean Average Precision) of different setup.

![Yolov8 Architecture](image)

**Fig. 2: Yolov8 Architecture**

**Table 2: Analysing mAP for different setup**

<table>
<thead>
<tr>
<th>Setup</th>
<th>Model</th>
<th>epochs</th>
<th>batch</th>
<th>Initial Learning Rate</th>
<th>Final Learning Rate</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup 1</td>
<td>yolov8s.pt</td>
<td>300</td>
<td>16</td>
<td>0.0001</td>
<td>0.001</td>
<td>86.5</td>
</tr>
<tr>
<td>Setup 2</td>
<td>Pretrained Model</td>
<td>300</td>
<td>16</td>
<td>0.0001</td>
<td>0.0001</td>
<td>86.0</td>
</tr>
<tr>
<td>Setup 3</td>
<td>yolov8s.pt</td>
<td>300</td>
<td>16</td>
<td>0.0001</td>
<td>0.0001</td>
<td>85.8</td>
</tr>
<tr>
<td>Setup 4</td>
<td>yolov8s.pt</td>
<td>300</td>
<td>16</td>
<td>0.001</td>
<td>0.001</td>
<td>91.7</td>
</tr>
<tr>
<td>Setup 5</td>
<td>Pretrained Model</td>
<td>200</td>
<td>32</td>
<td>0.01</td>
<td>0.01</td>
<td>91.6</td>
</tr>
<tr>
<td>Setup 6</td>
<td>yolov8s.pt</td>
<td>200</td>
<td>16</td>
<td>0.01</td>
<td>0.01</td>
<td>92.0</td>
</tr>
</tbody>
</table>
Highest mAP has been achieved from Setup 6. Table 3 displays the full training results, which provide insight into the model’s performance during the training process using Setup 6.

Table 3: Training Results

<table>
<thead>
<tr>
<th>Class</th>
<th>Images</th>
<th>Instances</th>
<th>BoxP</th>
<th>R</th>
<th>mAP50</th>
<th>mAP50-95</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>36</td>
<td>1111</td>
<td>0.895</td>
<td>0.903</td>
<td>0.92</td>
<td>0.637</td>
</tr>
<tr>
<td>0</td>
<td>36</td>
<td>41</td>
<td>1</td>
<td>0.934</td>
<td>0.993</td>
<td>0.73</td>
</tr>
<tr>
<td>1</td>
<td>36</td>
<td>85</td>
<td>0.864</td>
<td>0.835</td>
<td>0.886</td>
<td>0.602</td>
</tr>
<tr>
<td>2</td>
<td>36</td>
<td>72</td>
<td>0.844</td>
<td>0.899</td>
<td>0.88</td>
<td>0.644</td>
</tr>
<tr>
<td>3</td>
<td>36</td>
<td>43</td>
<td>0.947</td>
<td>0.977</td>
<td>0.993</td>
<td>0.729</td>
</tr>
<tr>
<td>4</td>
<td>36</td>
<td>59</td>
<td>0.886</td>
<td>0.966</td>
<td>0.967</td>
<td>0.689</td>
</tr>
<tr>
<td>5</td>
<td>36</td>
<td>48</td>
<td>0.974</td>
<td>0.938</td>
<td>0.981</td>
<td>0.703</td>
</tr>
<tr>
<td>6</td>
<td>36</td>
<td>62</td>
<td>0.878</td>
<td>0.952</td>
<td>0.923</td>
<td>0.631</td>
</tr>
<tr>
<td>7</td>
<td>36</td>
<td>71</td>
<td>0.979</td>
<td>1</td>
<td>0.993</td>
<td>0.744</td>
</tr>
<tr>
<td>8</td>
<td>36</td>
<td>56</td>
<td>0.921</td>
<td>0.875</td>
<td>0.863</td>
<td>0.621</td>
</tr>
<tr>
<td>9</td>
<td>36</td>
<td>51</td>
<td>0.933</td>
<td>0.941</td>
<td>0.97</td>
<td>0.73</td>
</tr>
<tr>
<td>division</td>
<td>36</td>
<td>43</td>
<td>0.862</td>
<td>0.884</td>
<td>0.837</td>
<td>0.627</td>
</tr>
<tr>
<td>equal</td>
<td>36</td>
<td>47</td>
<td>0.879</td>
<td>0.83</td>
<td>0.816</td>
<td>0.488</td>
</tr>
<tr>
<td>{</td>
<td>36</td>
<td>49</td>
<td>0.902</td>
<td>0.878</td>
<td>0.934</td>
<td>0.654</td>
</tr>
<tr>
<td>)</td>
<td>36</td>
<td>42</td>
<td>0.765</td>
<td>0.776</td>
<td>0.85</td>
<td>0.51</td>
</tr>
<tr>
<td>minus</td>
<td>36</td>
<td>75</td>
<td>0.953</td>
<td>0.81</td>
<td>0.941</td>
<td>0.356</td>
</tr>
<tr>
<td>multiplication</td>
<td>36</td>
<td>43</td>
<td>0.928</td>
<td>0.907</td>
<td>0.894</td>
<td>0.625</td>
</tr>
<tr>
<td>plus</td>
<td>36</td>
<td>50</td>
<td>0.914</td>
<td>1</td>
<td>0.98</td>
<td>0.704</td>
</tr>
<tr>
<td>{</td>
<td>36</td>
<td>46</td>
<td>0.759</td>
<td>0.935</td>
<td>0.877</td>
<td>0.648</td>
</tr>
<tr>
<td>)</td>
<td>36</td>
<td>47</td>
<td>0.8</td>
<td>0.767</td>
<td>0.864</td>
<td>0.646</td>
</tr>
<tr>
<td>[</td>
<td>36</td>
<td>40</td>
<td>0.89</td>
<td>0.95</td>
<td>0.928</td>
<td>0.617</td>
</tr>
<tr>
<td>]</td>
<td>36</td>
<td>41</td>
<td>0.915</td>
<td>0.902</td>
<td>0.942</td>
<td>0.675</td>
</tr>
</tbody>
</table>

3.6 Localization

Our model is designed to excel in recognizing handwritten characters. To achieve optimal results, it is important to preprocess the images prior to feeding them into the model.

Input Image Preprocessing

– Convert raw image to grayscale image.
– Covert the grayscale image into binary image (Black background and white characters).
Get output from trained YOLOv8 model The model returns a list of boundary box information of the detected objects. Boundary box information contains the coordinate of top-left, bottom-right, class and confidence rate.

Feed the output list to Localization model We have developed a localization model to analyse the bounding box information.

Algorithm 1: Localization Algorithm

Input: detection result: an array of tuples where each tuple contains (class label, center point, [x1, y1, x2, y2])

Output: Lines: an array of arrays containing the indices of bounding boxes that form a line

Sort detection result according to y-value of center_point and put them inside sorted_detection_result;

d_arr ← [];
foreach element in sorted_detection_result do
    d_i ← y_i - y_{i-1};
    Insert d_i into d_arr;
end
th ← Threshold(d_arr);
Line ← [];
foreach element in sorted_detection_result do
    d_i ← y_i - y_{i-1};
    if d_i < th then
        Insert i-th bounding box into Line;
    end
    else
        Sort the elements of Line according to x-value (in non-descending order);
        Print Line;
        Clear Line;
        Insert i-th bounding box into Line;
    end
end

Setting Threshold for line splitting Our model successfully detects the center point coordinates of all objects, including digits and signs. However, since we did not use an image-processing model for line detection, we developed a mathematical model to address this issue. We calculated the distance between neighboring objects based on their center y values and determined that if the distance exceeded a certain threshold value, the objects belonged to different lines.
We have created an array where bounding boxes are sorted according to their center y value.
Now, from the sorted array, we measure the height-distance between every $i^{th}$ and $(i+1)^{th}$ bounding box.
Then, from the distances, outliers are measured. The mean value of the outliers is the Threshold.

$$IQR = Q_3 - Q_1$$

Outlier Threshold = $Q_3 + 1.5 \times IQR$

$drr_{out} = \text{Values in } d_{arr} \text{ greater than or equal to Outlier Threshold}$

$$\text{Threshold} = \frac{1}{n} \sum_{i} drr_{out}$$

Here, $Q_i$ is the $i^{th}$ Quartile and $n$ is the number of data point.

4 Experimental Analysis

4.1 Data Preprocessing

In our research, we applied a two-step preprocessing approach, where we first converted raw images to grayscale and then binarized them into black and white before training our model. This approach was chosen to reduce the dimensionality of the image. We also explored training our model using raw images, but found that the predictions were poor.

4.2 Experimental Settings

We have tried with different setups as explained in 3.5. Finally, our model was trained using YOLOv8 for 200 epochs with a batch size of 16 images. We utilized an SGD optimizer with an initial learning rate of 0.01, final learning rate 0.01, SGD momentum 0.937 and optimizer weight decay 0.0005.

4.3 Result Analysis

From the test set of images we have found the results show in Table 4.

Analysis of the test results

Set 1 and 2: Prediction was fully accurate. No redundant or false objects were detected.
<table>
<thead>
<tr>
<th>Set</th>
<th>Original</th>
<th>Inverse Binarized</th>
<th>Detected</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>![Image 1](150x300 to 234x665)</td>
<td>![Image 2](235x300 to 319x665)</td>
<td>![Image 3](320x300 to 404x665)</td>
<td>Threshold: 65.32</td>
</tr>
<tr>
<td></td>
<td>![Image 1](150x300 to 234x665)</td>
<td>![Image 2](235x300 to 319x665)</td>
<td>![Image 3](320x300 to 404x665)</td>
<td>Converted Text:</td>
</tr>
<tr>
<td></td>
<td>![Image 1](150x300 to 234x665)</td>
<td>![Image 2](235x300 to 319x665)</td>
<td>![Image 3](320x300 to 404x665)</td>
<td>${(\sqrt{3}/2)\times2}/(8x2)$</td>
</tr>
<tr>
<td></td>
<td>![Image 1](150x300 to 234x665)</td>
<td>![Image 2](235x300 to 319x665)</td>
<td>![Image 3](320x300 to 404x665)</td>
<td>$=\sqrt{3}/2$</td>
</tr>
<tr>
<td></td>
<td>![Image 1](150x300 to 234x665)</td>
<td>![Image 2](235x300 to 319x665)</td>
<td>![Image 3](320x300 to 404x665)</td>
<td>$\Rightarrow$</td>
</tr>
<tr>
<td></td>
<td>![Image 1](150x300 to 234x665)</td>
<td>![Image 2](235x300 to 319x665)</td>
<td>![Image 3](320x300 to 404x665)</td>
<td>$\Rightarrow$</td>
</tr>
</tbody>
</table>

Table 4: Image Processing and Output Text

Set 3 and 4: More than one bounding boxes on one object are detected. This issue has been solved by following the steps below:

Say, there are two neighboring bounding box bbox1 and bbox2

We are checking how much area of b1 is inside b2 and vice versa.
If more than a threshold-percentage of bbox1 is inside bbox2 (or vice versa) then the one with more confidence rate is picked up. From several trial and error analysis we have found it feasible that the percentage of threshold-overlap should be 80%.

The final text-output of set 3 and 4 in Table 4 has been brought out by using above steps.
Mathematical method for finding overlap between two bounding box

Let \( P_1 = (x_{A1}, y_{A1}) \) and \( P_2 = (x_{A2}, y_{A2}) \) be the top left and bottom right points of bbox1 Also, let \( Q_1 = (x_{B1}, y_{B1}) \) and \( Q_2 = (x_{B2}, y_{B2}) \) be the top left and bottom right points of bbox2

Then, \( LX = \begin{cases} x_{A2} & \text{if } x_{A2} < x_{B2} \\ x_{B2} & \text{otherwise} \end{cases} \) \( RX = \begin{cases} x_{A1} & \text{if } x_{A2} > x_{B2} \\ x_{B1} & \text{otherwise} \end{cases} \)
\( LY = \begin{cases} y_{A2} & \text{if } x_{A2} < x_{B2} \\ y_{B2} & \text{otherwise} \end{cases} \) \( RY = \begin{cases} y_{A1} & \text{if } x_{A2} > x_{B2} \\ y_{B1} & \text{otherwise} \end{cases} \)

\( X_{\text{over}} = \begin{cases} RX - LX & \text{if } RX - LX > 0 \\ 0 & \text{otherwise} \end{cases} \) \( Y_{\text{over}} = \begin{cases} RY - LY & \text{if } RY - LY > 0 \\ 0 & \text{otherwise} \end{cases} \)

\( IA = X_{\text{over}} \times Y_{\text{over}} \)

\( area_A = (x_{A2} - x_{A1}) \times (y_{A2} - y_{A1}) \)

\( Overlapping = \begin{cases} \frac{IA}{area_A} & \text{if } area_A = 0 \\ 0 & \text{otherwise} \end{cases} \)

5 Result and Discussion

We have found the following confusion matrix from our own test set. We found average mAP 92.0%. If the model can be train more dataset, the accuracy will be higher.
6 Conclusion

Developing a reliable Bengali handwritten character recognition and localization system is a challenging task, but important for improving OCR technology for better communication and accessibility for Bengali speakers. Our proposed approach utilizes the YOLOv8 object detection algorithm for accurately detecting and localizing multiple Bengali digits and operators in an image. YOLOv8 returns localization data, such as the position of each object within the image, which is essential for identifying the line where the digit or sign belongs. By analyzing the position of each character, our method can convert handwritten mathematical equations into digital text, improving accessibility and efficiency for tasks such as educational and scientific writing.

References