Automatic Multiple Choice Question Evaluation Using Tesseract OCR and YOLOv8

Saikat Mahmud
Department of Computer Science
American International University-Bangladesh (AIUB)
Dhaka, Bangladesh
19-41556-3@student.aiub.edu

Kawshik Biswas
Department of Computer Science
American International University-Bangladesh (AIUB)
Dhaka, Bangladesh
20-42118-1@student.aiub.edu

Api Alam
Department of Computer Science
American International University-Bangladesh (AIUB)
Dhaka, Bangladesh
19-40880-2@student.aiub.edu

Rifat Al Mamun Rudro
Department of Computer Science
American International University-Bangladesh (AIUB)
Dhaka, Bangladesh
23-93113-3@student.aiub.edu

Nusrat Jahan Anannya
Department of Computer Science
American International University-Bangladesh (AIUB)
Dhaka, Bangladesh
anannya@aiub.edu

Israt Jahan Mouri
Department of Computer Science
American International University-Bangladesh (AIUB)
Dhaka, Bangladesh
israt.mouri@aiub.edu

Kamruddin Nur
Department of Computer Science
American International University-Bangladesh (AIUB)
Dhaka, Bangladesh
kamruddin@aiub.edu

***Abstract*— This paper presents a novel approach for automating the grading of multiple-choice question (MCQ) answer sheets using computer vision and pattern recognition techniques. The system examines student's marked answer sheet images by comparing with the question sheet image and answer keys. The computer vision and pattern recognition helps extracting pertinent data such as question number detection, MCQ option detection and the answer markings. The proposed approach reliably produces the output report that displays the students' correct answers with an accuracy of 0.98 F1 score and 0.99 mAP from any form or unstructured question script. This approach can provide a dependable and effective grading system, reducing manual work and offering prompt feedback to students without any constraints on the answer sheets.**

***Index Terms*—Optical character recognition (OCR), image processing, object detection, YOLOv8**

I. Introduction

The evaluation and grading of Multiple-Choice Question (MCQ) answer sheets present difficulties in the educational environment. Teachers often face the dilemma of either manually reviewing MCQ answer sheets or relying on MCQ evaluators such as [1] [2] or using high-end Optical Mark Recognition (OMR) scanners that necessitate adherence to a template. Besides, many general OMRs cannot detect two-response answers or some even reject the input image when no choice is selected for a given question [3]. The formatting rules for conventional approaches such as the use of uniform symbols [4], fixed formatted sheets [5], or precisely aligning the answer boxes, often demand rigorous adherence. Students may face challenges when using such templates, leading to errors in the answering process. Unintentional markings on incorrect options or answering the wrong questions could occur due to the use of separate answer spaces or sheets. To provide a more inclusive and flexible method of assessment, this paper attempts to create a system that can automatically read and evaluate multiple choice answer sheets, regardless of any fixed formatted template or separate sheet. The suggested system would use an object detection method to evaluate and interpret the answers provided by examines. The system will be able to distinguish between valid and incorrect responses, and duplicates, and give a count of correct answers on each answer sheet. By eliminating the need for strict formatting guidelines, the system will accommodate two marking annotations, i.g. filled circles and crosses. This study has the potential to significantly improve the efficiency, usability, and adaptability of the multiple-choice answer sheet evaluation process by utilizing object detection and OCR technology. This paper will examine the system's design, implementation, and evaluation while demonstrating how well it automates the marking of

979-8-3503-5409-6/24/$31.00 ©2024 IEEE
DOI 10.1109/CAI59869.2024.00054

multiple-choice answer sheets.

The overall contributions of this paper are summarized as follows:

1) Training of a custom object detection model based on YOLOv8
2) Integration of Optical Character Recognition (OCR) and object detection in the system
3) Validation, testing, and manually comparing the proposed system's accuracy with the actual outputs

The rest of the paper is organized as follows: the related works are described in Section II, the methodology of the proposed system is presented in Section III, the Section IV explains the dataset, the training procedure, and parameter settings, the Section V explains in details the system workflow, the experimental results are presented in Section VI, the limitations and future works are outlined in Section VII, and finally, we conclude the paper in Section VIII.

II. Related Works

Various authors have proposed numerous evaluation methodologies. Most of these approaches rely on Optical Mark Recognition (OMR) or transoptic papers, bubble sheets, fixed templates, or image registration processes.

Sattayakawee [3] discussed a system that first scans the grid answer sheet and converts it to a grayscale image. The image is then processed using projection profile and thresholding techniques to detect the lines and grids of the answer sheet. Then, it segments each question and determines the selected choice by analyzing the local projection profile and applying thresholding to separate the cross-mark from the lines for fetching answers. With high accuracy, the system follows some constraints, e.g., wrong answers should be erased cleanly, no cross-out is allowed, and all markings should be uniform. Fisteus el at. [4] presented a system called Eyegrade for automatically grading multiple choice exams. The system uses a regular webcam for mark recognition from a continuous stream and optical character recognition of handwritten student identification numbers using OpenCV, Hough transform, TRE and Pygame. The tool has been validated with a set of experiments that show the ease of use, the reduction in grading time, and an increase in the reliability of results compared to conventional, more expensive systems. The limitation of their approach is that the resolution and image quality of a regular webcam can limit its application in a real academic context, and the geometry of the answer table is known prior. Muangprathub et al. [5] proposed an approach utilizing image processing techniques and template matching. The system operates 2.5 times faster than the conventional manual method and supports any pencils or pens used on thin papers and low-cost gridded paper that is easy to use in a typical test. Their approach has very low accuracy in cases with incomplete markings, such as small, overflowed, deleted, or unclean markings.

Alomran and Chai [6] proposed an image processing system and finder pattern algorithm to preprocess scanned test papers. It uses a fixed-templates sheet for each specific portion. The system was implemented with MATLAB, and the Ricoh Aficio MP C5501A was used as the scanning device. Depending on optimal conditions, although the system has a high accuracy it requires at least 40% of the test areas to be annotated and could not process images that were titled more than 6 degrees in a clockwise or anticlockwise direction. Afifi and Hussain [7] proposed a technique where they used an image registration technique to extract the answer boxes (ROI) from answer sheets and train a machine learning classifier to recognize the class of each answer box (i.e., confirmed, crossed out or blank answer). They present a dataset including six real MCQ assessments with different answer sheet templates and evaluate two strategies of classification: a straightforward approach (i.e., crossed out and empty) and a two-stage classifier approach (i.e., crossed out and confirmed). They test two handcrafted feature methods and a convolutional neural network. Shaikh et al. [8] proposed an image processing and handwritten pattern recognition approach using Convolutional Neural Networks (CNN) in a written answer-based evaluation, i.e., options A/B/C/D. The system was implemented using Raspberry PI, SSH server, MATLAB for segmentation, and Python for CNN models.

Hassan et al. [9] proposed Optical Character Recognition (OCR) to extract and segment characters from exam papers and then use Natural Language Processing to score the exam papers academically. Trained with 50 samples of numeral sets (0-9) an accuracy of 81% was achieved with images taken from various light and paper conditions. Chai [10] proposed an image processing algorithm using a finder pattern and projective transformation technique in a specified answer sheet, which was implemented using MATLAB R2014b. It needs 352 seconds to mark, annotate, and save 100 answer sheets. The system can not grade any sheets if their orientation is distorted, the multiple-choice circles are not filled, or if the papers are not properly aligned. Zhang [11] proposed an approach for marking objective exam questions. The approach involves projecting the special answer sheet of the examination paper several times to split out each option from each line. The number of white dots in the circle is counted, and the largest number is considered as the chosen answer. The approach uses image segmentation. Jocovic et al. [12] have used computer vision algorithms for the automated assessment of pen and paper tests. The implemented computer vision system consists of four modules: the Zoning module, the Scanning module, the Processing module, and the Verification module. The authors have used the Hough transform algorithm for circle recognition. The approach requires a pre-established format for the test, which may not be feasible for all types of exam sheets. Rasiq et al. [13] developed a mobile-based system using a bubble sheet template. Their methodology involved an image-slicing technique, considering the number of columns and rows of the circles. The circles were marked as selected based on the number of black pixels within them, surpassing a predefined percentage threshold. The system varies in accuracy and may generate wrong results if the image is noisy. Karunanayake [14] proposed an OMR sheet evaluation system

in which a predefined region containing all correct answers is manually marked and isolated as a template image. The template image is then matched with the corresponding area on the OMR sheet, resulting in cropping a region with identical characteristics. Both the reference template and the cropped image are converted into binary images and subjected to blob filtering, facilitating the identification of the correct number of answers. Nguyen et al. [15] proposed a system that uses captured images of the OMR answer sheets, which are then processed using techniques such as Hough transform, skew correction, normalization, and tick mark recognition.

III. Methodology

We evaluated the most promising state-of-the-art related work methods and found that OCR and object detection methods can be used together for better accuracy of any structured or unstructured MCQ question marking. Our analysis revealed that Tesseract OCR outperformed other OCR engines and YOLOv8 performed better than other object detectors in detecting fill, cross, or other user markings on MCQ question scripts. The object detection and OCR methods are discussed in subsequent subsections. The workflow diagram of the proposed system is presented in Figure 1.

A. Object Detection Model

As previously discussed in Section III, the object detection model identifies the markings of selected answers in MCQ papers. It provides the bounding box coordinates of those markings for further processing. Given our objective to employ an object detection approach, selecting an appropriate object detection model became a pivotal consideration in this case. YOLO (You Only Look Once) was chosen, guided by the rationale that it has been acclaimed in multiple research papers [16] for its lightweight nature, commendable speed, and accuracy. Following the documentation provided by *Ultralytics* , YOLOv8 emerged as a preferable choice due to its attributes as a fast, accurate, and user-friendly model, succeeding its predecessors and improved architecture.

B. OCR Engine

The system extracts, rearranges, and stores the question number's coordinates using the OCR engine. Then the selected answer text is extracted by cropping each answer portion with the help of bounding box coordinates provided by the YOLO model and stored systematically. Among the multiple Optical Character Recognition (OCR) engines available, Tesseract OCR [17] was selected for the system. Tesseract is renowned for its robust capabilities, providing high customization with multilingual support [18] for extracting text from document images. Several authors including Ramiah et al. [19], Dangiwa and Kumar [20] have developed applications for both Android and iOS platforms using Tesseract OCR. This indicates the adaptability of the OCR engine for mobile devices. Besides, Robby et al. [21], Niharika et al. [22] demonstrated the efficacy of Tesseract OCR in supporting multilingualism. Being free, open-source, available as an SDK, multi-language support, and demonstrating relatively good accuracy in text extraction, it stands out compared to other OCR engines that have been benchmarked in various papers [23].

IV. Data Training

A. Image Collection

We collected data, focusing on 250 student-answered MCQ script images associated with a specific course within our university. These images encompassed two distinct types of markings: (i) selected or answered and (ii) crossed out. The selected markings corresponded to filled circles, indicating a chosen answer by the student. In contrast, the crossed-out marking featured a filled circle with a cross (X) in the center, signifying an initial erroneous selection that the student subsequently corrected.

B. Dataset

For model training, those two marking categories were designated as the 'fill' and 'cross' classes, respectively. Among the 250 images, 200 images were randomly chosen for training and validation purposes. The rest of the 50 images were kept for testing. Since the dataset was small, 85% of the images were allocated for training, while the remaining 15% were reserved for validation.

TABLE I: Dataset Descriptive Information

Class Names	Train		Validation	
	Images	Instances	Images	Instances
Fill	170	2610	30	447
Cross	170	657	30	108
All	170	3267	30	557

C. Image Prepossessing

A manual examination of each image was conducted. Subsequently, specific preprocessing steps were applied as deemed necessary. These steps encompassed corrections such as rotation adjustment to ensure proper alignment, fine-tuning of brightness levels, and the extraction of Region of Interest (ROI) through cropping. These measures were implemented to enhance the overall quality and suitability of the images for subsequent stages of analysis and model training.

V. Implementation of the Proposed System

A. Preprocessing Techniques

1) Rotation Correction: To correct the image rotation (if needed) the *image_to_osd()* function from *pytesseract* [24] is invoked and based on the angle information the image is rotated through the *imutils* library.

2) Cropped to Text Area: The image is then processed to isolate the region of interest (ROI) containing the relevant textual content while excluding any extraneous margin areas. Initially, the input image is duplicated to preserve the original data. Subsequently, the image is converted to grayscale to simplify the analysis. Gaussian blur is then applied to reduce noise and smoothen the image. A binary inverse thresholding technique employing Otsu's method highlights the text and markings against the background. To enhance the accuracy of the text area extraction, horizontal lines are eliminated

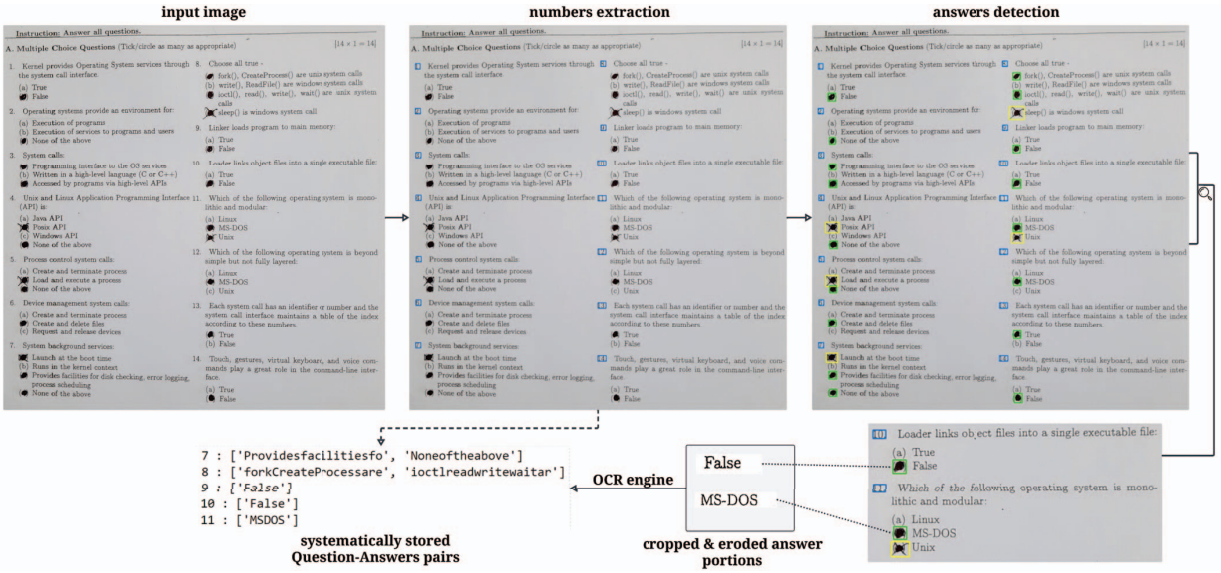

Fig. 1: System Workflow Diagram

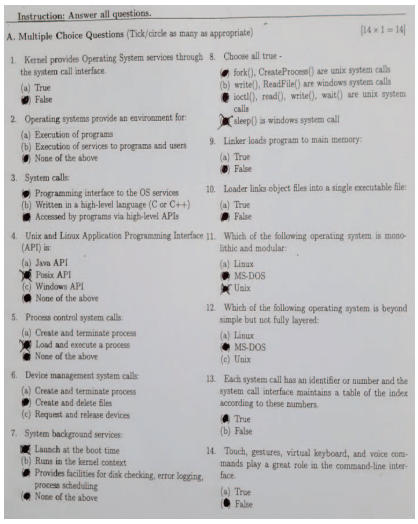

Fig. 2: Sample Image of the MCQ Question Script

Fig. 3: Instances of 'fill' class

Fig. 4: Instances of 'cross' class

from the thresholded image through morphological opening using appropriate kernels. The first set of kernel dimensions (30,1) is used to eliminate any interference caused by these lines, ensuring accurate identification and isolation of the text area. The second set of kernel dimensions (5,26) is applied to dilate the thresholded image. Dilation aids in merging separate contours into a single contour, further refining the delineation of the text area. The contours of the resultant image are then identified, and the largest contour is selected to represent the text area. By extracting the bounding rectangle of this contour, the text area is effectively isolated from the rest of the image and stored as the original image for any future steps.

3) Getting Eroded Image: Later, we focused on preparing the image obtained from the previous step for efficient Optical Character Recognition (OCR) processing. After converting it to grayscale, a median blur is applied to mitigate noise and irregularities present in the grayscale image, and then Otsu's method for the binarization. To optimize OCR engine performance further, morphological erosion is employed using a predefined kernel of dimensions (3,3) for refining the boundaries of the text characters, which provides a cleaner and more distinct representation for the OCR engine.

```
Mark achieved: 8
Correct answers: [3, 4, 5, 6, 8, 10, 11, 12]
Wrong answers: [1, 2, 9, 13, 14]
No answers: [7]
```

Fig. 5: Marks Calculation Output Sample

B. Getting the Question Numbers and Their Coordinates

The Tesseract OCR engine is invoked by calling the *image_to_data()* function from *pytesseract* within the eroded im-

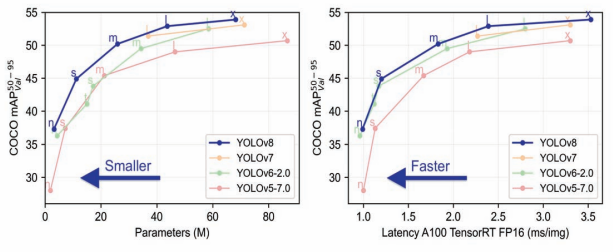

Fig. 6: YOLOv8 Performance Comparisons with Previous Versions [25]

age. The function was configured by passing two parameters, *oem 3* and *psm 4*. The *psm 4* configuration is mandatory to prevent the OCR engine from automatically segmenting the image and force it to consider column data and concatenated row-wise text. The OCR output was structured as a dictionary, containing data such as text, confidence scores, positions, and dimensions of recognized elements. The extracted data are then filtered to retain only the pertinent OCR data. To this end, regular expressions are used to isolate specific OCR data that match a predefined pattern, in that case, one or two digits followed by a dot (.), typically representing question numbers in MCQs. We also ensured that recognized elements exhibited a confidence score exceeding a threshold, enhancing the reliability of the extracted data.

C. Sorting The OCR Data

In the next step, we focus on organizing the extracted OCR data in a structured and intuitive manner. This task involves invoking a specialized function designed to sort the data coordinates in a column-wise arrangement, prioritizing left-to-right and top-to-bottom order. To initiate this organization, previously filtered OCR data for pertinent question numbers is passed to the function. Additionally, the width of the image is conveyed as a parameter which is a key determinant for achieving a column-wise arrangement. To facilitate proper column-wise segmentation, the image width is halved and a fixed value is subtracted to define an appropriate range for the x-axis, ensuring accurate partitioning into two distinct columns. After this OCR extraction step, we moved on to the YOLO model prediction.

D. YOLO Model's Prediction

A defined function is called for YOLO model prediction and bounding box generation. This step plays a pivotal role in identifying the markings within the MCQ question paper. To commence this step, the function receives the image, which is then duplicated to preserve the original data for visualization. A predefined set of labels is associated with respective class IDs to aid in interpreting the model's predictions. The pre-trained YOLO model is loaded for prediction, leveraging the model's ability to detect distinct classes, i.e., 'cross' and 'fill.' The model's predictions yield bounding boxes corresponding to the detected elements. These bounding boxes, along with their respective class IDs are systematically processed to determine the location and type of each detected element.

Further, a distinction is made based on the class ID: if the class ID is not '0' (representing a 'cross'), the bounding box coordinates are appended to a selection list. Visual markers, such as rectangles, are drawn around the identified elements to facilitate verification and validation. To ensure a coherent order and arrangement, the selection list is passed to the same sorting function as discussed previously, along with a third parameter to execute the process as YOLO bounding box coordinates. The function does the same but this time it returns a sorted single list of bonding box data named 'YOLO_data'.

E. Getting The Answers

After that, we employ a predefined 'get_answer()' function to extract MCQ question numbers and their corresponding answers from the extracted OCR data and the YOLO bounding box coordinates. This process is a fundamental component of our automated MCQ question marking system. To commence this step the function is invoked with the thresholded image, 'YOLO_data' from the preceding step, and the 'ocr_data' as parameters. Inside the function, the YOLO bounding boxes are transformed from the *xyxy* format to *xywh* format using the built-in function *xyxy2xywh()* from the *ultralytics* library. The OCR engine is then invoked to extract answer strings, utilizing the configuration parameter to ensure the text is considered as a uniform block. The extracted text data is then filtered to retain only alphanumeric characters, limiting the selection to the initial 20 characters for improved later assessment. All those procedures are repeated for the student-answered images and then compared with the stored question-answer pair, fetched from the teacher-answered images.

F. Comparing Answers and Marks Calculation

To compare and get marks for an individual student another function is called. The function is designed to operate with two sets of answers, teacher answers and student answers. It performs a detailed analysis of these answer strings to compute the student's marks by identifying wrong, correct, and unanswered question numbers. To achieve this, several variables are initialized, including the student's marks, to keep track of the student's total marks and lists for unmatched answers, wrong answers, correct answers, and unanswered questions. The function iterates the teacher's answers and checks them against the corresponding student's answers for a specific question number (key) in the student's answer list. If it does, the function compares the student's answer(s) against the correct answer(s) provided by the teacher. If the student's answer does not exactly match any of the correct answers, the tuple index corresponding to this unmatched answer is recorded. Depending on the comparison results, the function updates the tracking variables accordingly. The question number is added to the wrong answers list if the student's answer is incorrect or contains unmatched answers. If the student's answer matches all correct answers for that question, the student's marks are increased by 1, and the question number

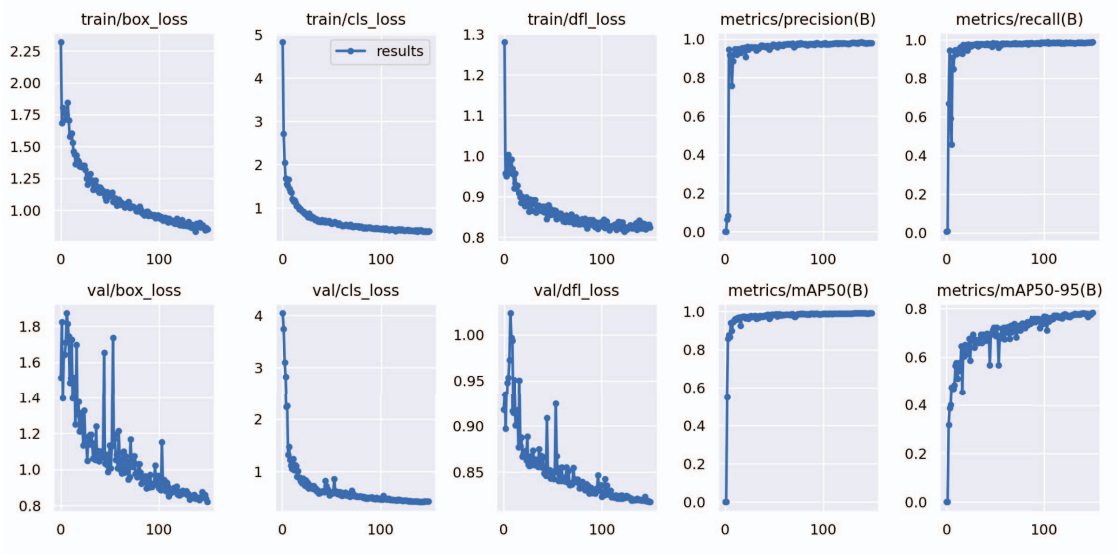

Fig. 7: Training Results of the Proposed Model

is added to the correct answer list. The question number is included in the not-answered list if the student did not answer a particular question. The implementation of this system can be found in the GitHub repository[1].

VI. Results and Discussion

Since there are only two classes and our implementation should be on mobile devices, the model was trained to employ YOLOv8n (nano) architecture. The YOLOv8n has 3.2 million parameters and a speed of 8.4ms on COCO dataset outperforming the previous versions as shown in Figure 6. Hyperparameter tuning was done by considering the dataset and object types as shown in Table II. Each instance of the 'fill' and 'cross' classes was treated as an object to be detected. Training of the model was performed on the Google Colab environment. After training the model, it demonstrates remarkable performance. With a confidence threshold of 0.585, the model attains an impressive F1-score of 0.98, shown in Figure 8 followed by the calculation of precision and recall using Equations (1) and (2). This signifies that the model excels in both precision and recall when considering bounding box predictions with confidence scores exceeding the specified threshold.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

Additionally, it attains a recall score of 0.988, demonstrating its capability to retrieve all relevant objects accurately. Furthermore, the model's overall performance is underscored by a mean Average Precision (mAP) score of 0.99. The mAP score serves as a comprehensive metric, offering an evaluation of the model's effectiveness across various object classes. This performance assessment is measured using an Intersection over Union (IoU) threshold of 0.5 mentioned in Table III, reflecting the model's robustness in object detection and classification.

[1]https://github.com/SaikatMahmud/mcq_evaluation_with_ocr_yolov8

TABLE II: Training Hyperparameters

Training Hyperparameters	Details
Epoch	150
Image size	1280 x 1280
Batch size	10
Learning rate (lr0, lrf)	0.01

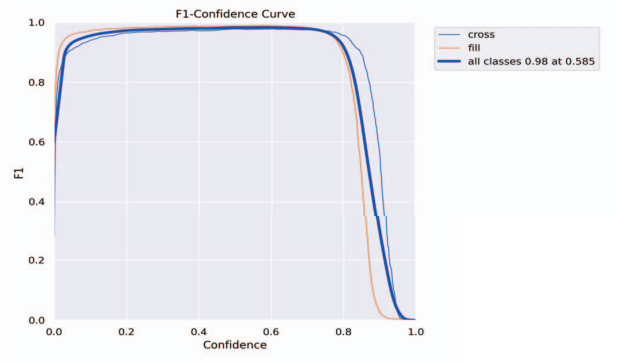

Fig. 8: F1 vs Confidence curve

TABLE III: Trained Model Summary

Class	Precision	Recall	mAP50	mAP50-95
Fill	0.984	0.994	0.991	0.754
Cross	0.974	0.982	0.99	0.81
All	0.979	0.988	0.99	0.782

The remaining set of 50 images was tested which took an average execution time of 3.5 seconds per script in Core i7, 8th Generation intel processor with 8GB RAM. The examination revealed only two instances where the system erroneously classified 'cross' instances as 'fill,' attributing this misclassification to the small size of the crosses that fell below the model's detection threshold. The error counts from the test are presented in Table IV.

TABLE IV: Error Count of 50 MCQ Script Test Dataset

Category	Erroneous count
fill	0
cross	2
Letter (OCR)	10
Undetected question number (OCR)	4
Wrongly detected question number (OCR)	3

VII. Limitation and Future Work

The system may encounter challenges with low-quality input images. For example, in the situation of tilted, hazy, or faint printed text. The system may erroneously identify false positive objects as true positives outside of designated MCQ areas, which are identified as the limitations and future works.

VIII. Conclusion

The paper introduces an innovative approach to streamline the Multiple-Choice Question (MCQ) evaluation process by integrating Optical Character Recognition (OCR) and object detection methods. While the proposed methods exhibit certain limitations, their potential for acceptance within the educational community, comprising both educators and students, remains noteworthy. A key advantage is that there is no longer a need for a specific template paper, which improves readability and ease of use for educators and learners alike. Despite these merits, there is room for future enhancements which are outlined in the limitation and future work section. In conclusion, ongoing efforts for refinement and optimization can position this methodology as a valuable tool in the educational assessment landscape.

References

[1] H. E. Ascencio, C. F. Peña, K. R. Vásquez, M. Cardona, and S. Gutiérrez, "Automatic multiple choice test grader using computer vision," in *2021 IEEE Mexican Humanitarian Technology Conference (MHTC)*. IEEE, 2021, pp. 65–72.

[2] B. Kommey, E. Keelson, F. Samuel, S. Twum-Asare, and K. K. Akuffo, "Automatic multiple choice examination questions marking and grade generator software," *IPTEK The Journal for Technology and Science*, vol. 33, no. 3, pp. 175–189, 2022.

[3] N. Sattayakawee, "Test scoring for non-optical grid answer sheet based on projection profile method," *International Journal of Information and Education Technology*, vol. 3, no. 2, p. 273, 2013.

[4] J. A. Fisteus, A. Pardo, and N. F. García, "Grading multiple choice exams with low-cost and portable computer-vision techniques," *Journal of Science Education and Technology*, vol. 22, pp. 560–571, 2013.

[5] J. Muangprathub, O. Shichim, Y. Jaroensuk, and S. Kajornkasirat, "Automatic grading of scanned multiple choice answer sheets," *International Journal of Engineering and Technology (UAE)*, vol. 7, no. 2, pp. 175–179, 2018.

[6] M. Alomran and D. Chai, "Automated scoring system for multiple choice test with quick feedback," *International Journal of Information and Education Technology*, vol. 8, pp. 538–545, 2018.

[7] M. Afifi and K. F. Hussain, "The achievement of higher flexibility in multiple-choice-based tests using image classification techniques," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 22, no. 2, pp. 127–142, 2019.

[8] E. Shaikh, I. Mohiuddin, A. Manzoor, G. Latif, and N. Mohammad, "Automated grading for handwritten answer sheets using convolutional neural networks," in *2019 2nd International conference on new trends in computing sciences (ICTCS)*. Ieee, 2019, pp. 1–6.

[9] S. Hassan, J. J. Smile, K. Karthick, R. R. R. Goplan, P. P. Damodharan, and D. N. Suguna, "Answer sheet evaluation using ai," Apr 2019. [Online]. Available: https://ijsrst.com/IJSRST196231

[10] D. Chai, "Automated marking of printed multiple choice answer sheets," in *2016 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*. IEEE, 2016, pp. 145–149.

[11] D. Zhang, "The algorithm of objective questions marking system based on projection method," in *2017 6th International Conference on Computer Science and Network Technology (ICCSNT)*. IEEE, 2017, pp. 463–466.

[12] V. Jocovic, M. Marinkovic, S. Stojanovic, and B. Nikolic, "Automated assessment of pen and paper tests using computer vision," *Multimedia Tools and Applications*, pp. 1–22, 2023.

[13] G. R. I. Rasiq, A. Al Sefat, and M. F. Hasnain, "Mobile based mcq answer sheet analysis and evaluation application," in *2019 8th International Conference System Modeling and Advancement in Research Trends (SMART)*, 2019, pp. 144–147.

[14] N. Karunanayake, "Omr sheet evaluation by web camera using template matching approach," *International Journal for Research in Emerging Science and Technology*, vol. 2, no. 8, pp. 40–44, 2015.

[15] T. D. Nguyen, Q. Hoang Manh, P. Bui Minh, L. Nguyen Thanh, and T. M. Hoang, "Efficient and reliable camera based multiple-choice test grading system," in *The 2011 International Conference on Advanced Technologies for Communications (ATC 2011)*, 2011, pp. 268–271.

[16] S. Srivastava, A. V. Divekar, C. Anilkumar, I. Naik, V. Kulkarni, and V. Pattabiraman, "Comparative analysis of deep learning image detection algorithms," *Journal of Big data*, vol. 8, no. 1, pp. 1–27, 2021.

[17] R. Smith, "An overview of the tesseract ocr engine," in *ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 629–633. [Online]. Available: https://storage.googleapis.com/pub-tools-public-publication-data/pdf/33418.pdf

[18] R. Smith, D. Antonova, and D.-S. Lee, "Adapting the tesseract open source ocr engine for multilingual ocr." in *MOCR '09: Proceedings of the International Workshop on Multilingual OCR*, ser. ACM International Conference Proceeding Series, V. Govindaraju, P. Natarajan, S. Chaudhury, and D. P. Lopresti, Eds. ACM, 2009, pp. 1–8. [Online]. Available: https://storage.googleapis.com/pub-tools-public-publication-data/pdf/35248.pdf

[19] S. Ramiah, T. Y. Liong, and M. Jayabalan, "Detecting text based image with optical character recognition for english translation and speech using android," in *2015 IEEE Student Conference on Research and Development (SCOReD)*, 2015, pp. 272–277.

[20] B. A. Dangiwa and S. S. Kumar, "A business card reader application for ios devices based on tesseract," in *2018 International Conference on Signal Processing and Information Security (ICSPIS)*, 2018, pp. 1–4.

[21] G. A. Robby, A. Tandra, I. Susanto, J. Harefa, and A. Chowanda, "Implementation of optical character recognition using tesseract with the javanese script target in android application," *Procedia Computer Science*, vol. 157, pp. 499–505, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050919311640

[22] G. L. Niharika, S. Bano, P. S. Kumar, T. Deepika, and H. Thumati, "Character recognition using tesseract enabling multilingualism," in *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 2020, pp. 1321–1327.

[23] A. P. Tafti, A. Baghaie, M. Assefi, H. R. Arabnia, Z. Yu, and P. Peissig, "Ocr as a service: an experimental evaluation of google docs ocr, tesseract, abbyy finereader, and transym," in *Advances in Visual Computing: 12th International Symposium, ISVC 2016, Las Vegas, NV, USA, December 12-14, 2016, Proceedings, Part I 12*. Springer, 2016, pp. 735–746.

[24] S. Hoffstaetter, "Pytesseract: A Python wrapper for google tesseract," Aug. 2022. [Online]. Available: https://pypi.org/project/pytesseract/

[25] Ultralytics, "Yolov8." [Online]. Available: https://docs.ultralytics.com/models/yolov8/