International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 04 | Apr 2022 www.irjet.net p-ISSN: 2395-0072

Online Exam Proctoring using Deep Learning

Jai Pawar[1], Jyoti Prasad[1], Prathamesh Shanbhag[1], Charmi Chaniyara[2]

[1]Dept. of Information Technology, Atharva College of Engineering, Maharashtra, India
[2]Assistant Professor, Dept. of Information Technology, Atharva College of Engineering, Maharashtra, India

---

Abstract - *The Covid-19 pandemic has affected the educational system worldwide, leading to the near-closures of schools, universities, and colleges. This has resulted in increased demand for E-learning, and the subsequent growth in the remote learning industry. With E-learning, exams are also undertaken remotely. Research suggests most examinees hold the perception that it is easier to subvert the system in online exams than in traditional ones. To ensure the smooth running of exams, Online Exam Proctoring will be essential, where it would ensure fair, unbiased proctoring of exams. The goal of the project is to be able to develop a website that will track examinee activity during the exam to prevent any malpractice.*

***Key Words*: SSD, coco, proctoring, EfficientNet**

1. INTRODUCTION

This paper proposes a system to address and mitigate the difficulties faced by proctors during the execution of examinations online. The proposed system is a non-invasive software, that automates routine remote exam proctoring tasks. It is in the form of a website, which allows examinees to attempt the exam with their webcams switched on. The system tracks examinee presence, mobile phone presence, presence of more than one individual, and book presence in the webcam feed using Deep Learning techniques, and immediately warns the examinee if any of the above are found. The Deep Learning approach used for object detection is a Single Shot Detector (SSD), which was pre-trained on the coco dataset. The teacher can schedule tests via google forms links by inputting them to the website, and the examinees can access active test links from the attempt test section.

2. LITERATURE REVIEW

In [1] Nigam, A., Pasricha, R., Singh, T. *et al.* reviews more than 40 research articles on automated proctoring software and present drawbacks of earlier methods as well as advantages of newer methods. It served as a guide to keeping track of advancements in this sub-field with updated info, considering it was published in 2021.

In [2] Y. Atoum, L. Chen, A. X. Liu, S. D. H. Hsu and X. Liu proposed a system, which employs two cameras to record the examinee from two different angles, along with a microphone to record audio data. Data obtained from the above input devices are used for gaze detection and speech recognition. The approach helped us as a reference for creating our system.

In [3] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, et al compares the Speed and accuracy of multiple popular modern convolutional networks, specifically object detection networks. The metric selected to evaluate the above models was mAP, while speed was calculated in milliseconds. The findings from this paper helped us finalize an object detection model that best fits our use case.

In [4] Liu, W. et al proposed a method for detecting objects in images using a single deep neural network, the Single Shot Detector. SSD is simple when compared to existing methods that require object proposals, like RCNN because it completely eliminates proposal generation and subsequent pixel or feature resampling stages and encapsulates all computation in a single network. As a result of this, SSD is simple to train and integrate into systems that require a detecting component.

3. ANTI CHEATING MEASURES

Common cheating methods observed after an anonymous survey of examinees were:

1. Copying questions in text format and pasting them into a group chat in another window.
2. Capturing an image of the questions via a mobile phone and circulating it to a group of co-conspirators.
3. Referencing books/study material from in front of the computer screen itself.
4. Moving away from the computer screen to reference books/study material.
5. Inviting co-conspirators to attempt the test on the same laptop/machine.

Measures undertaken by our software to mitigate/prohibit the above cheating methods:

1. Mandatory enforced Fullscreen mode:

   Prohibits the examinee from opening another tab or window, eliminating Section 3, point 1.

2. Examinee detection every 1000 milliseconds:

   Prohibits the examinee from moving away from the test screen, mitigating Section 3, point 4.

3. Multiple person detections every 1000 milliseconds:

   Throws a warning if the website detects multiple individuals attempting the test along with the examinee on the same machine, eliminating Section 3, point 5.

4. Mobile phone detection:

   Throws a warning if a mobile phone is detected, eliminating Section 3, point 2.

5. Book detection:

   Throws a warning if a book is detected, eliminating Section 3, point 3.

4. SYSTEM DESIGN

4.1 Architecture

The remote testing system comprises a website, whose front-end is created using React.js, and the database and backend are created using Firebase.

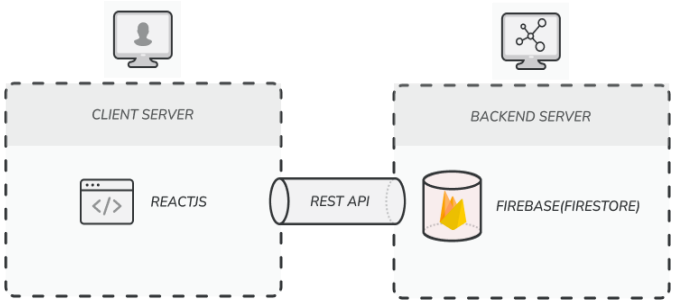

Fig 1: System architecture diagram

The front end has a home page and an attempt exam page. The home page describes the proctoring measures undertaken by the website, and the dos and don'ts. It has two buttons on the home page, offering two functionalities:

1. "Schedule test" button allows teachers to upload their tests to the website. The test is supposed to be in the form of a standard MCQ-based format, whose link is to be entered in an URL input field, along with the test name. The form link for the test and the test's name will then be saved into our database.

2. "Attempt test" button allows the examinee to attempt scheduled tests. It leads to an available tests page, which has a list of test names and their links displayed. Examinees can attempt the concerned test, which opens the test in a new window.

4.2 Testing Procedure

The test window opens in full screen mode, which disables any attempts at switching tabs or switching windows. If the user exits full screen mode, the test window closes, and the test terminates. This feature also makes sure that the examinee cannot cheat by sharing a screenshot of the test screen with third parties while the test is active, as he cannot change tabs even if he captures a screenshot. This was achieved by manipulating the JavaScript window object.

The examinee can now attempt the test, while his webcam is on the entire time. The examinee must stay within the field of view of the webcam, as the object detection model runs inference on the webcam feed in real-time. The website sends a warning using the react-toastify library if either of the following objects is detected in the webcam feed:

a. Book

b. Mobile phone

c. Multiple people

It also fires a warning using the react-toastify library, if a singular person is not detected. This is to ensure that the examinee does not leave the laptop unattended while the test is underway.

Detection of mobile phones via the camera makes sure that the examinee cannot take a photograph of the test screen.

Detection of multiple people and subsequent warnings makes sure that multiple people cannot give the test on the same laptop or help the examinee in real-time.

If any of these warnings stack up to 3 times, the test automatically terminates, and the examinee will be routed back to the home screen, along with a message as to why the test was terminated.

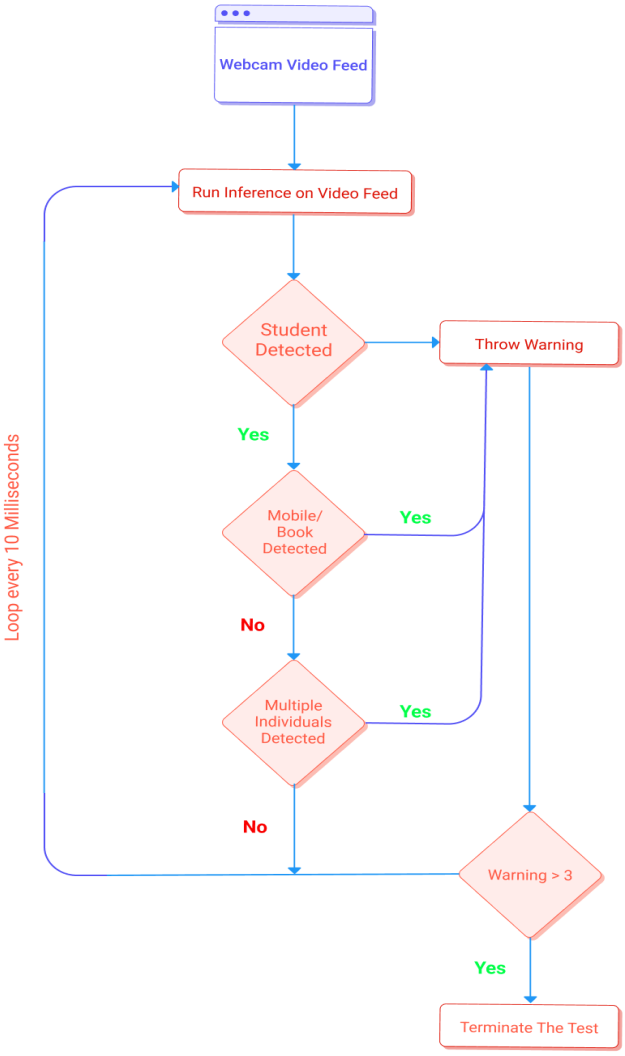

Fig 2: System working flowchart

5. COMPARATIVE ANALYSIS

Evaluation metrics chosen to compare object detection models are Mean Average Precision (mAP) and speed of prediction.

Mean Average Precision is calculated by taking an average of Average Precision (AP). Average Precision is a way to summarize the precision-recall curve into a single value, which represents the average of all precisions.

The higher the value of mAP, the more accurate the object detection model is in detecting classes.

Speed of prediction can be defined by the amount of time taken to make a prediction (in milliseconds). Lower the amount of time taken, the higher the speed of prediction.

Comparison of multiple popular object detection algorithms based on their mAP and speed of prediction:

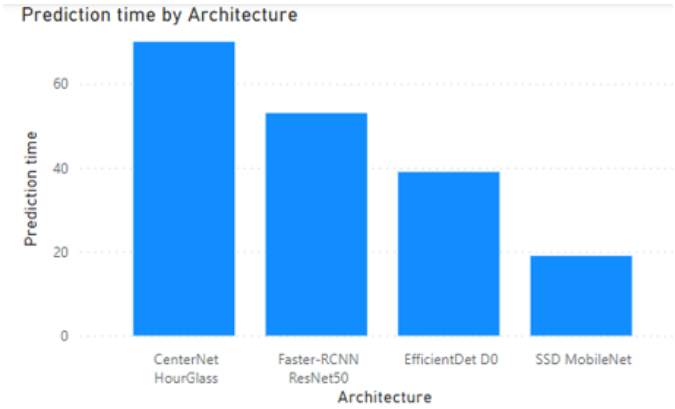

Fig 3: Prediction time vs architecture

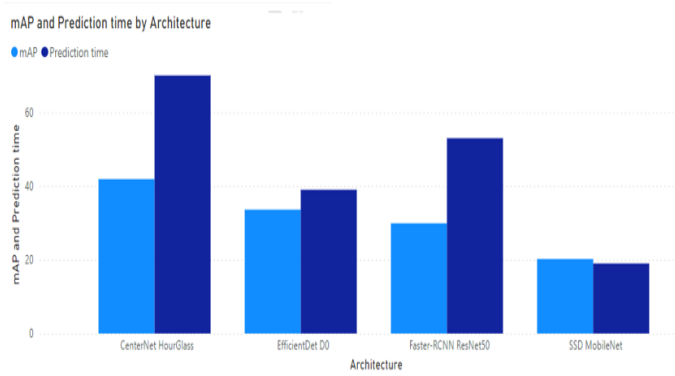

Fig 4: mAP and Prediction time vs Architecture

These are benchmark precision and speed values of respective architectures, sourced from the TensorFlow model garden and from [3].

Our test case requires a model that can predict in real-time, hence the speed of prediction is a priority, while mAP can be modest. After referencing [3], which laid out speed vs accuracy tradeoffs of multiple popular architectures, as well as referencing the official benchmark speed/mAP metrics of TensorFlow's implementations, we selected the SSD object detector, due to its relatively high speed of prediction.

6. TEST CASES

6.1 Unit testing:

Tested the front end, backend, and object detector individually, as well as together.

6.2 Load testing:

Accessed the website from multiple accounts at the same time to check if it crashes and created multiple assessments in quick succession.

6.3 UI testing:

Testing responsiveness on multiple resolution sizes.

6.4 Functional testing:

Testing each button routes to the correct page.

7. RESULT

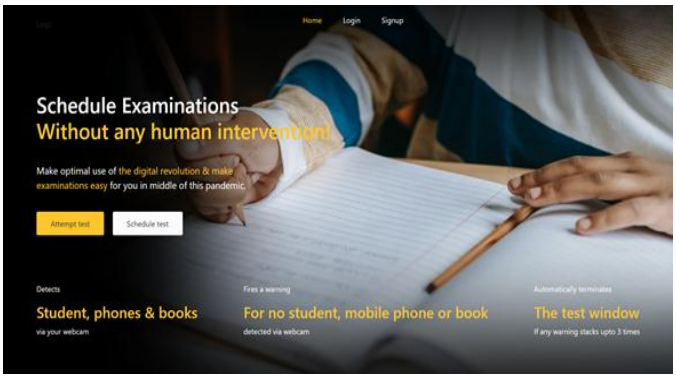

Fig 5: The website's landing page

Describes the proctoring measures in place, as well as the Attempt test and Schedule test buttons.

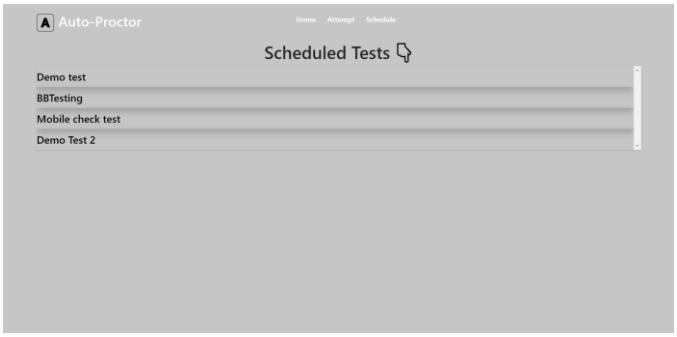

Fig 6: Available tests page

Displays tests scheduled by teachers and ready to be undertaken by the examinees.

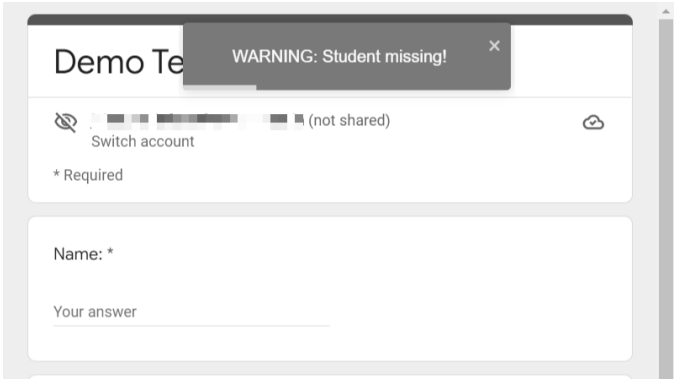

Fig 7: Examinee Missing warning

Displays a warning issued as soon as the examinee moves away from the webcam field of view.

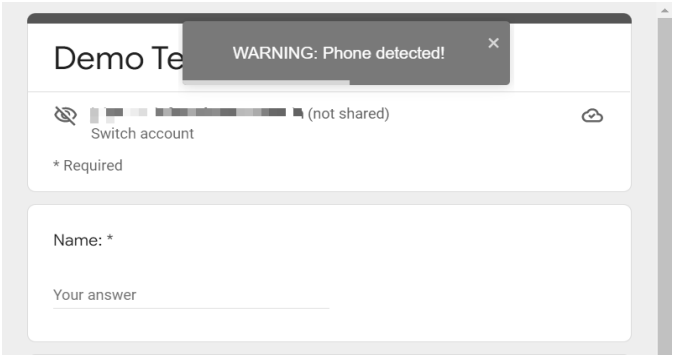

Fig 8: Phone detected warning

Displays a warning issued as soon as a mobile phone is detected in the webcam's field of view.

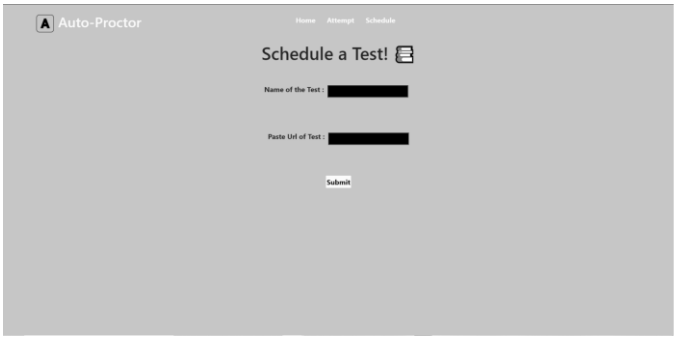

Fig 9: The schedule tests page

Allows teachers to schedule tests, by entering the test's name & MCQ-Based assessment link.

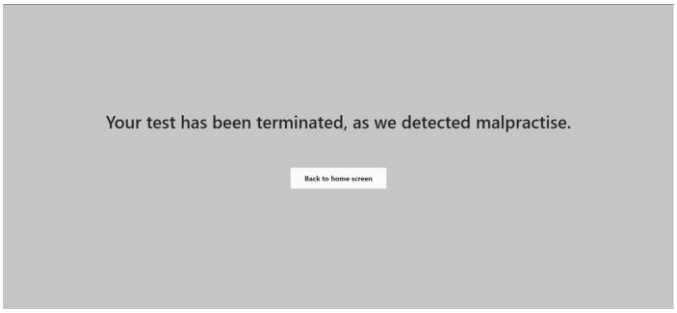

Fig 10: Test terminated page

This page is displayed when any of the above warnings stack up to 3 times & returns a home screen button.

8. CONCLUSION

Rather than an end-to-end solution to remote proctoring, eliminating human proctors completely, this project is intended to be an asset to human proctors' toolkit for efficient and non-invasive remote proctoring.

The project obviates the need of downloading third-party software, doesn't store user data, doesn't require constant invigilation, and facilitates smoother remote exam scheduling and undertaking. It automates routine and mundane proctoring tasks, reducing pressure on human proctors and reducing bias and human error.

REFERENCES

[1] Nigam, A., Pasricha, R., Singh, T. et al. A Systematic Review on AI-based Proctoring Systems: Past, Present and Future. Educ Inf Technol 26, 6421–6445 (2021). https://doi.org/10.1007/s10639-021-10597-x

[2] Y. Atoum, L. Chen, A. X. Liu, S. D. H. Hsu and X. Liu, "Automated Online Exam Proctoring," in IEEE Transactions on Multimedia, vol. 19, no. 7, pp. 1609-1624, July 2017, doi: 10.1109/TMM.2017.2656064.

[3] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi et al "Speed Accuracy trade-offs for modern convolutional object detectors", in https://arxiv.org/abs/1611.10012

[4] Liu, W. et al. (2016). SSD: Single Shot MultiBox Detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science(), vol 9905. Springer, Cham. https://doi.org/10.1007/978-3-319-46448-0_2