Real-Time Gaze and Object Recognition System for Resource-Constrained Settings: Intelligent Malpractice Detection in CBT Documentation

By:

Philip-Toye Joshua (AI\ML Engineer)

jimidev4@gmail.com

Abdulraheem Balikis (AI/ML Engineer)

abdulraheempelumi@gmail.com

Introduction

The Intelligent Malpractice Detection System (IMDS) represents a lightweight, offline proctoring solution designed for real-time detection of misconduct during computer-based examinations (CBT). Utilizing a webcam and a low-power laptop, IMDS integrates gaze tracking and object detection technologies to efficiently identify instances of malpractice. Nations in West Africa, including Nigeria, Ghana, and Liberia, are increasingly adopting CBT for national examinations; however, they contend with challenges such as insufficient human oversight, the unavailability of cost-effective video proctoring solutions, and prevalent instances of cheating facilitated by mobile phones. IMDS mitigates these issues by providing continuous gaze monitoring, contextual object detection for illicit instruments, and complete offline functionality, thereby enabling scalable and equitable testing for thousands of remote and rural examination centers. AI-based monitoring tools such as ProctorU and HonorLock are often costly, relying on subscription licenses and cloud infrastructure that are better suited to environments with robust connectivity and funding—conditions not commonly available in many West African educational settings. These systems are primarily designed for Western institutions and do not align well with the realities of schools and test centers operating under resource constraints.

Statement Of Problem

CBT exams in Africa face significant malpractice issues, particularly in low-budget or rural areas. Common problems include off-screen assistance, hidden phones, and unauthorized individuals. Most AI proctoring systems are unsuitable due to their high technical requirements (bandwidth, GPUs, cloud servers), which many African institutions lack.

Constraints

Table 1..1 .system constraints/bottlenecks

Type	Details
Hardware	Webcam; CPU(with or without GPU)
Latency	Real-Time interaction expected(<= 2s delay
Accuracy	System must balance false positive/negatives

Design Choices

The development of this system prioritizes a highly modular and flexible architecture, coupled with carefully selected models and a precise activation logic, to ensure reliable and efficient performance in detecting and responding to user gaze deviations.Modular Architecture: Enhancing Flexibility and Maintainability The system is broken down into distinct, specialized modules, each responsible for a specific aspect of the overall functionality. This modular approach significantly improves the system's flexibility, allowing for independent development, testing, and potential future upgrades or replacements of individual components without impacting the entire system.

Gaze Detection Module (MediaPipe + TensorFlow): This core module is responsible for real-time prediction of the user's gaze direction. It leverages the power of MediaPipe, a comprehensive framework for building multimodal applied machine learning pipelines, in conjunction with TensorFlow, an open-source machine learning platform. This combination provides a robust foundation for accurate and efficient gaze tracking.

Object Detection Module (YOLOv8n): This module is strategically designed to activate only when a significant deviation in user gaze is detected. It utilizes YOLOv8n (You Only Look Once, version 8 nano), a highly optimized and lightweight version of the YOLO family of object detection models. Its efficiency makes it ideal for real-time critical item detection, ensuring that the system can quickly identify relevant objects in the user's environment once their gaze shifts.

Logic Control Module: Serving as the central orchestrator, this module manages the

intricate interplay between the other components. Its responsibilities include:

Gaze State Management: Continuously monitoring and interpreting the output of the Gaze Detection Module to determine the current gaze state (e.g., focused, deviated).

Timing Control: Implementing precise timing mechanisms to ensure that the Object Detection Module is activated only after a sustained period of gaze deviation, preventing false positives from momentary shifts.

Event Logging: Recording significant events and system states, providing valuable data for analysis, debugging, and performance evaluation.

Logger: Dedicated to data persistence, the Logger module systematically stores all relevant detections. This includes:

1.Gaze Detections: Records of gaze direction, confidence levels, and timestamps.

2. Face Detections: Data related to the detected face, such as position and orientation, which can be crucial for gaze analysis.

3.Object Detections: Details of detected objects, including their bounding box coordinates, class, and confidence scores.

4.Data Formats: The logger supports both human-readable text logs for quick review and CSV (Comma Separated Values) format for easy programmatic analysis and integration with data visualization tools.

Model Selection: Optimizing for Performance and EfficiencyThe choice of machine learning models for both gaze and object detection was driven by a balance of accuracy, computational efficiency, and suitability for real-time applications.

- **Gaze Detection Model:**Uses a MobileNetV2 model to classify screen vs. off-screen gaze from cropped eyes.The data set used was the Columbia gaze data set which contains about 5000 plus images
- **Object Detection Model:** YOLOv8- small detects items like phones,laptops,books,people in real time.Activation Logic: Ensuring Robust and Intelligent Triggering

The system employs an activation logic to ensure that the object detection module is triggered only when there is a confident and sustained deviation in user gaze. This prevents spurious activations due to minor head movements or momentary distractions.

- **Rolling Buffer (Deque) for Smooth Predictions:** To mitigate the impact of transient fluctuations in gaze detection confidence, the system utilizes a rolling buffer, specifically a `deque` (double-ended queue). This data structure efficiently stores a recent history of gaze confidence scores. By analyzing a window of past predictions, the system can smooth out instantaneous variations and obtain a more stable and reliable assessment of gaze.
- **Conditional Triggering based on Gaze Confidence:** The object detection module is activated only when a specific condition is met: the average gaze confidence falls below a predefined threshold of 0.5 for a continuous period of at least 2 seconds.
    - **Average Gaze Confidence < 0.5:** This threshold indicates a significant dip in the system's confidence that the user is still focused. A value below 0.5 suggests a high probability of gaze deviation.
    - **≥ 2 Seconds Duration:** The requirement for the low confidence to persist for at least 2 seconds acts as a crucial filter, preventing the system from reacting to fleeting distractions or minor head movements. This duration ensures that the detected gaze deviation is sustained and likely indicative of a genuine shift in attention.

This robust activation logic is fundamental to the system's accuracy and user experience, minimizing false positives and ensuring that the object detection module is engaged only when genuinely necessary, thus optimizing resource utilization and providing a seamless interaction.

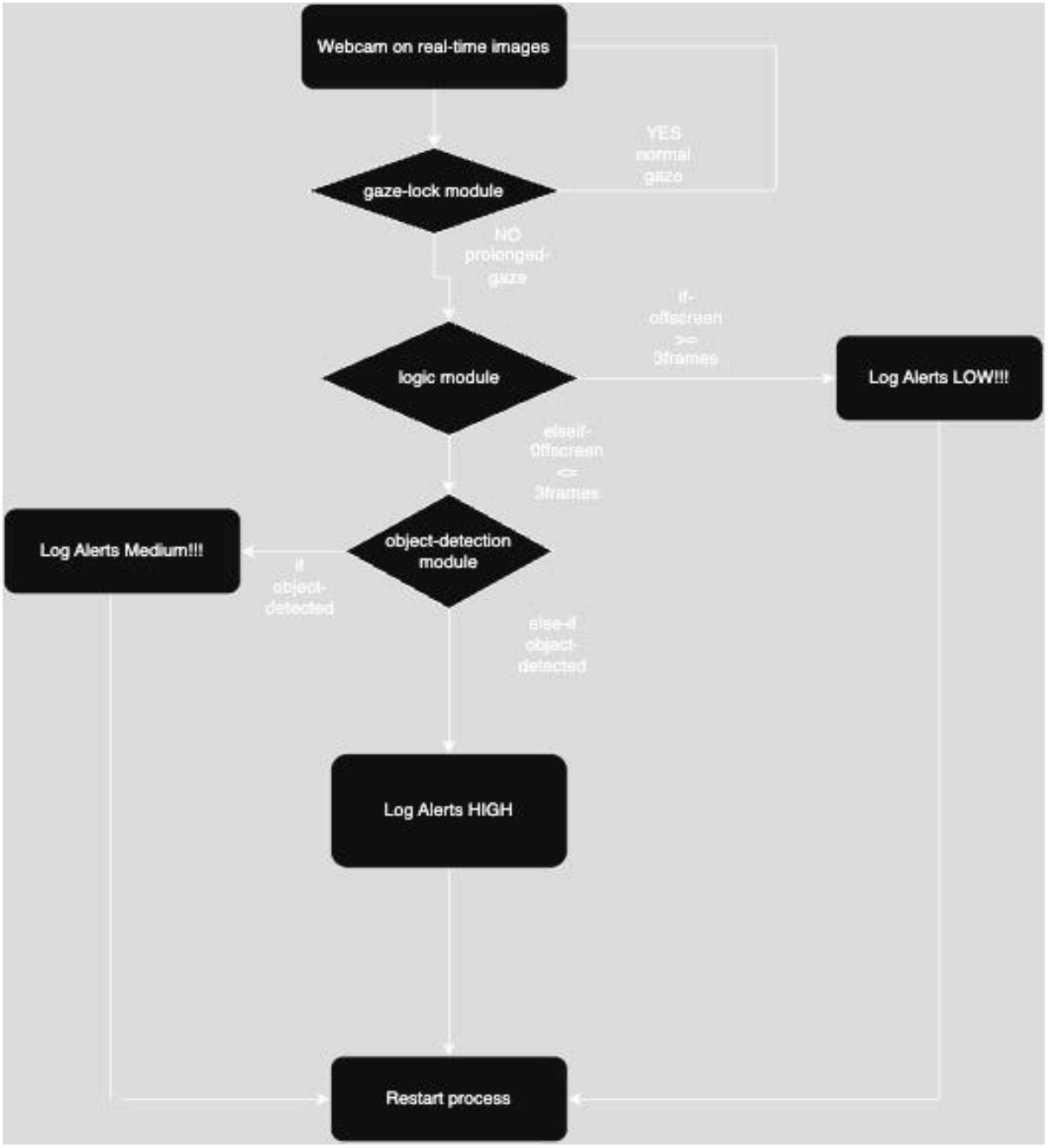

Fig 1.1 system flow chart

Technologies Used

The implementation of the Intelligent Malpractice Detection System (IMDS) was carried out using a combination of modern machine learning frameworks and computer vision libraries. The system was developed with a focus on compatibility with low-resource environments and cross-platform support.

Category	Specification
Programming Language	Python 3.9
MachineLearning Frameworks	TensorFlow (for gaze classification), Ultralytics YOLOv8 (for object detection)
Computer-Vision Libraries	OpenCV (image processing), MediaPipe FaceMesh (facial landmark detection)
Runtime Environment	CPU-only execution; no GPU required
Memory Requirements	Minimum 4GB RAM
Deployment Platforms	Cross-platform support: Windows, macOS (including M1), and Linux
Operational Mode	Fully offline; no internet connection required for inference or monitoring

This choice of technologies ensures the system remains lightweight, accessible, and functional in environments with limited computational power and no reliance on cloud infrastructure.

System Performance Metrics Evaluation

This table 4.6 summarizes key performance metrics for the real time system, seemingly in the context of malpractice event detection. Understanding these metrics is crucial for assessing the model's effectiveness and identifying areas for improvement.

Table 1.2 .system performance metrics

Metric	Value	Definition	Interpretation
Accuracy	54%	(TP + TN) / (TP + FP + TN + FN)	Moderately reliable; may suffer from frequent False Positives or False Negatives.
Precision	60%	TP / (TP + FP)	Good at avoiding false accusations (when it predicts positive, it's often correct).
Recall	42%	TP / (TP + FN)	Misses more than half of actual malpractice events. Needs significant improvement.
Specificity	65%	TN / (TN + FP)	Fairly good at correctly identifying innocent behavior (non-malpractice events).
F1 Score	49.5%	2 * (Precision * Recall) / (Precision + Recall)	Balances precision and recall. Reflects overall effectiveness, which is moderate

			here.

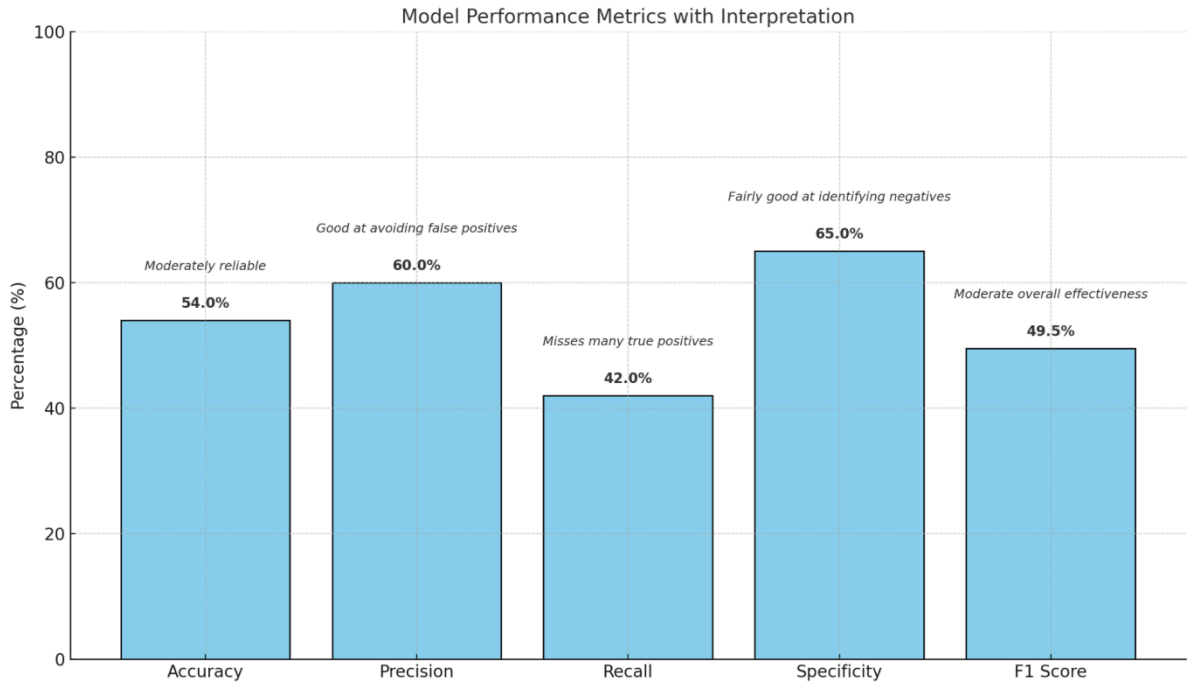

Fig 1.2 visual representation for System performance

Conclusion

This system offers an intelligent and modular foundation for automated CBT surveillance. By combining lightweight gaze tracking and contextual object detection, it enhances both usability and security with minimal compute cost.