Real-Time Malpractice Detection in Examination

Posham Navya
Department of Electronics and Electrical Engineering, IIT Guwahati
Email: posham@iitg.ac.in

***Abstract*—With the rise in remote education, the need for scalable, secure, and fair proctoring solutions has become increasingly important. This paper proposes a real-time malpractice detection system that leverages advanced computer vision and AI techniques to automate the proctoring process. The system integrates MediaPipe FaceMesh for precise face detection and head pose estimation, YOLOv8m for real-time object detection (focusing on identifying mobile phones), and a custom-trained CNN classifier to analyze student behaviours indicative of potential cheating. By combining these components, the proposed solution ensures continuous, objective, and scalable monitoring of examinees during examinations.**

***Index Terms*—Malpractice Detection, CNN, YOLOv8m, MediaPipe FaceMesh, Head Pose Estimation, Real-time Monitoring**

I. Introduction

With the increasing shift towards digital platforms for education and assessments, ensuring secure and fair examination environments has become crucial. Traditional human proctoring methods are limited in scalability and are prone to human error and fatigue, which can lead to oversight and compromise the integrity of examinations.

Advancements in computer vision and artificial intelligence (AI) offer effective solutions by automating the monitoring process. By leveraging camera feeds and machine learning models, these systems can detect suspicious behaviours in real-time with greater consistency and objectivity than human invigilators.

This project presents a real-time malpractice detection system that integrates multiple computer vision techniques. MediaPipe FaceMesh is used for precise face detection and head pose estimation to track student attention and detect abnormal movements. YOLOv8m is employed for object detection, particularly focusing on identifying mobile phones. Additionally, a custom-trained CNN model classifies student behaviours based on visual cues that may indicate malpractice. Together, these components form a robust, automated monitoring system aimed at enhancing fairness and integrity in digital assessments.

II. OBJECTIVES

1) Detect multiple faces in real-time
2) Estimate head pose direction (forward, left, right)
3) Detect mobile phones using YOLOv8m
4) Classify student activities using a custom CNN
5) Display real-time alerts and save logs to CSV every 5 seconds

III. WORKFLOW DIAGRAM

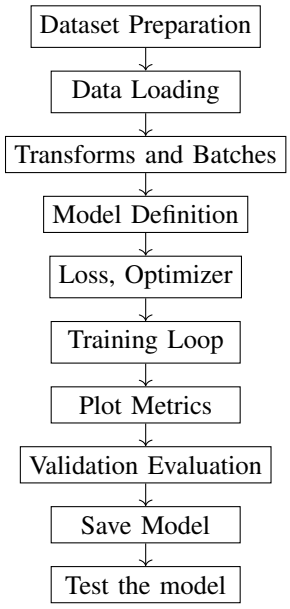

Fig. 1: CNN Model Training Flowchart

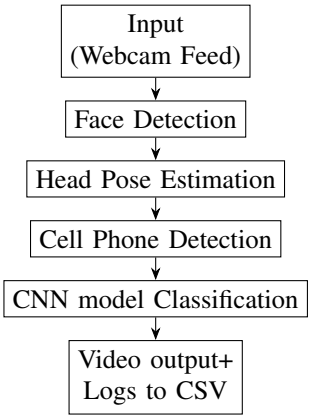

Fig. 2: System Workflow Diagram

IV. ABOUT DATASET

The dataset, sourced from Kaggle, consists of labeled webcam images representing different student behaviours such as normal act, cheating, giving object, looking friend and giving code. Images are organized in class-wise folders and preprocessed for consistency. The data is preprocessed by resizing, normalizing, and converting into tensors using torchvision

transforms. The images are split into training and validation sets with a 90:10 ratio. To address slight class imbalance, class weights are used during training. This dataset enables the CNN model to learn and classify different head orientations and suspicious behaviors effectively, forming the backbone of the action recognition module.

V. MODEL DESIGN

A. Custom CNN for Action Classification

Methodology: Convolutional Neural Networks (CNNs) are a class of deep learning models designed to process data with a grid-like structure, such as images. They automatically extract spatial hierarchies of features through convolutional layers, enabling them to detect patterns like edges, shapes, and complex objects. In this project, a CNN is used to classify webcam images of students into categories such as "looking friend", "cheating", or "normal act". The custom CNN model was designed with simplicity and real-time efficiency in mind. It leverages multiple convolutional blocks followed by pooling operations to reduce spatial dimensions and capture abstract features. These features are then passed to fully connected layers that map them to class probabilities. This architecture ensures high accuracy while maintaining inference speed suitable for live webcam input.

1) Custom Dataset Handling:

- A custom Dataset class was written to manage image loading.
- Automatically scans image folders organized by class names.
- Assigns integer labels to each class based on folder structure.
- Handles both training and test data loading with optional transformations.
- Uses PIL to load images and apply preprocessing like resizing and normalization via torchvision.transforms.

B. Model Architecture in PyTorch

1) Convolutional Layers with ReLU activation
2) MaxPooling Layers
3) Flattening into 1D vectors
4) Fully Connected Layers
5) Output layer produces raw class scores

VI. RESULTS OF CNN MODEL

After training the CNN over 25 epochs, the model demonstrated strong generalization and classification accuracy across multiple action classes. Training Performance:

1) The training loss reduced significantly in the early epochs and remained low throughout, while validation loss plateaued with minimal fluctuations, indicating good generalization.
2) Accuracy and F1 scores for both training and validation sets consistently approached 97–99confirming that the model learned to classify distinct head-based actions effectively.

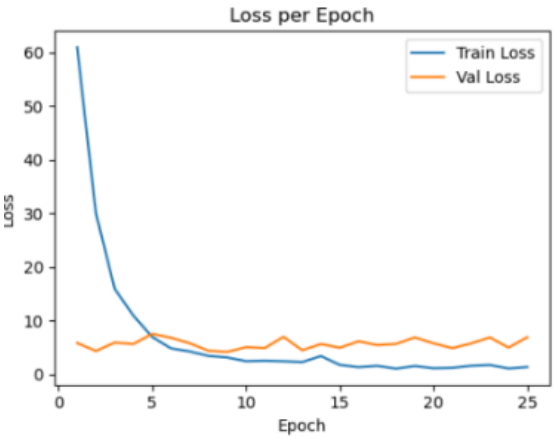

Fig. 3: Train vs Validation Loss

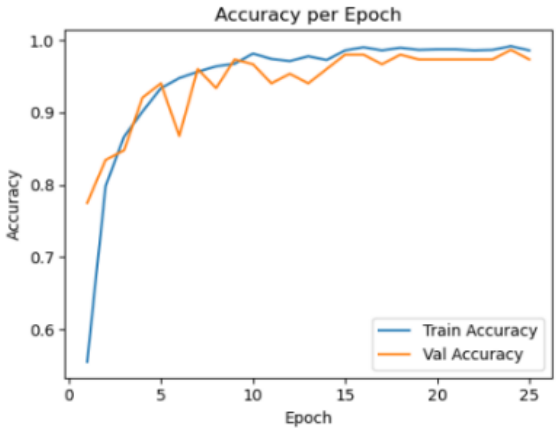

Fig. 4: Train vs Validation Accuracy

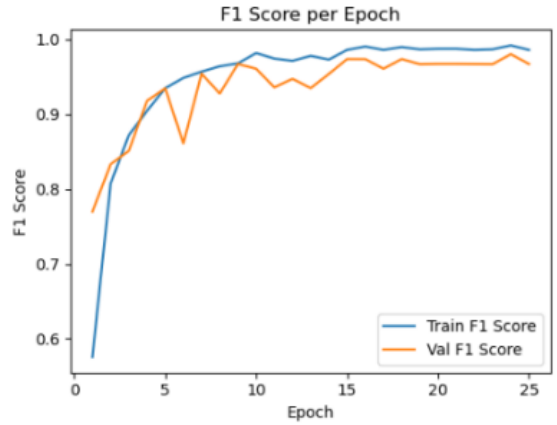

Fig. 5: Train vs Validation F1 score

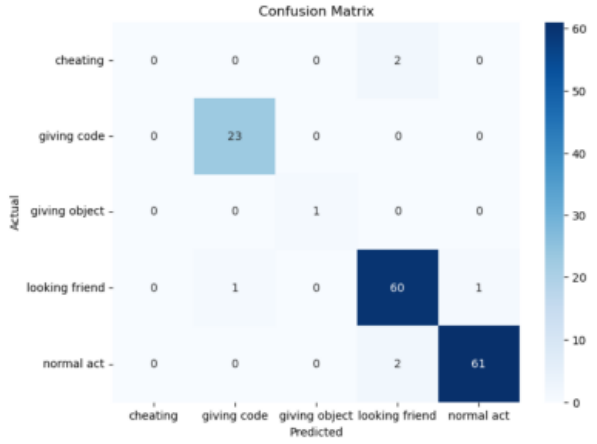

Fig. 6: Confusion Matrix

Classification Report:

	precision	recall	f1-score	support
cheating	0.00	0.00	0.00	2
giving code	0.96	1.00	0.98	23
giving object	1.00	1.00	1.00	1
looking friend	0.94	0.97	0.95	62
normal act	0.98	0.97	0.98	63
accuracy			0.96	151
macro avg	0.78	0.79	0.78	151
weighted avg	0.95	0.96	0.95	151

Fig. 7: Classification Report

VII. MEDIAPIPE AND YOLOV8M INTEGRATION

This project utilizes a combination of MediaPipe solutions and the YOLOv8m model to detect key elements contributing to malpractice behavior in a real-time video feed. These components complement the CNN-based action classification and serve as auxiliary modules to enhance detection accuracy

1) **Face Detection:** MediaPipe FaceMesh is used for realtime, lightweight face detection. It provides 468 3D facial landmarks for each detected face, offering precise tracking even under partial occlusions or angle changes. These landmarks are used as the basis for further analysis like head pose estimation.

2) **Head Pose Estimation:** By leveraging key landmarks from FaceMesh (such as the nose tip, eye corners, and chin), the system estimates head orientation. These landmarks are mapped to 3D model points and projected to 2D image space, allowing calculation of Euler angles (pitch, yaw, roll). Based on these angles, the system classifies head pose direction as ”Looking Left”, ”Looking Right”, or ”Looking Forward”.

3) **Phone Detection:** The pretrained YOLOv8m model is used to detect phones in the webcam feed. It was selected for its balance between speed and accuracy. When a phone is detected with confidence above a set threshold, an alert is displayed on the screen and logged.

Together, these models enhance the robustness of the system. FaceMesh ensures accurate multi-face detection, head pose adds behavioral context, YOLOv8m identifies cheating devices. All the components of the system—CNN-based classification, MediaPipe-based face detection and head pose estimation, and YOLOv8m for phone detection—are integrated into a unified real-time video processing pipeline. The webcam feed is first processed using MediaPipe FaceMesh, which identifies all visible faces and extracts 3D facial landmarks which are used to estimate head pose direction (e.g., looking left, right, or forward.

1) The webcam feed is first processed using MediaPipe FaceMesh, which identifies all visible faces and extracts 3D facial landmarks which are used to estimate head pose direction (e.g., looking left, right, or forward.
2) Each face region is cropped and passed through the trained CNN model, which classifies the current action or behavior of the student.
3) Simultaneously, YOLOv8m processes the same frame to detect the presence of mobile phones using bounding boxes and confidence thresholds.

All detections are visualized on-screen with bounding boxes, class labels, and real-time alerts to provide immediate feedback on suspicious activities. Additionally, detection events are systematically logged into a CSV file at fixed time intervals, enabling post-examination review and analysis. This seamless integration of visual feedback and event logging ensures that various behaviours and objects are monitored concurrently, resulting in a comprehensive and accurate real-time proctoring solution.

VIII. INPUT AND OUTPUT

Input: Live webcam feed
Output: Real-time bounding boxes and labels on screen as:

1) Face ID: to show number of faces
2) Head Pose: looking forward, left or right,
3) Action: to give the CNN prediction
4) Phone detected or not.
5) Alerts: Gives alerts if phone is detected or face is looking left or right for 5 sec, also gives warning if multiple faces are detected.
6) Alerts: All the alerts and warnings are also notified through sound or beep.

IX. OUTPUT RESULTS

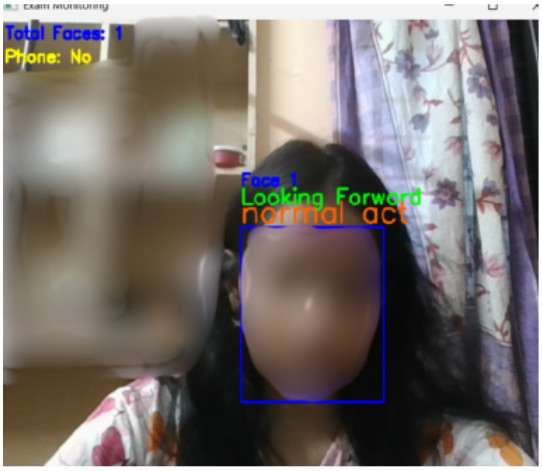

Fig. 8: Showing 'normal act' in the exam and the head looking forward.

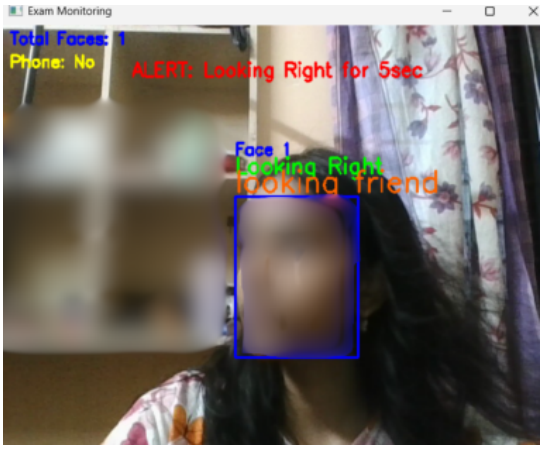

Fig. 9: Showing Alert as Looking Right for 5 sec in the exam and the head looking right and predicted as looking friend.

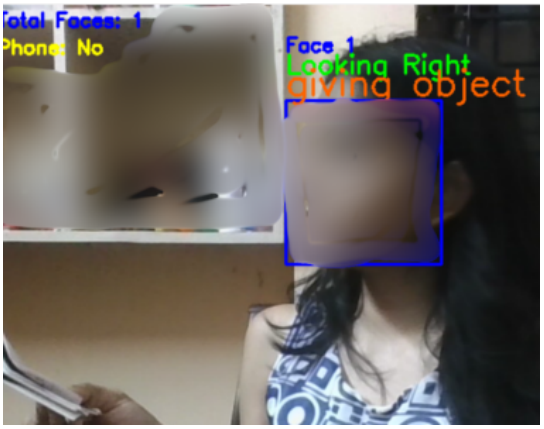

Fig. 10: Showing 'giving object' in the exam and the head looking right.

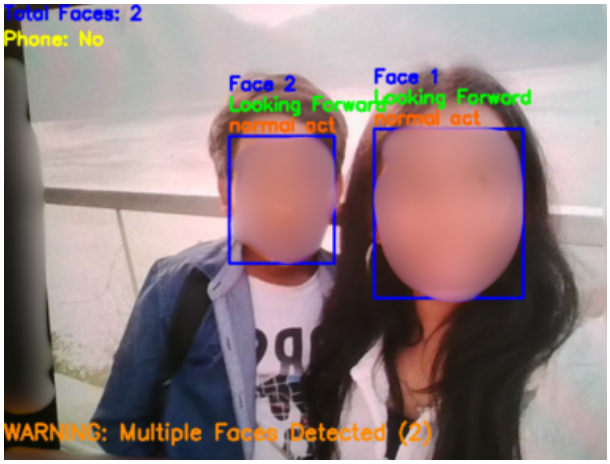

Fig. 11: Showing Warning as Multiple faces detected while also giving head pose and CNN output for every face detected.

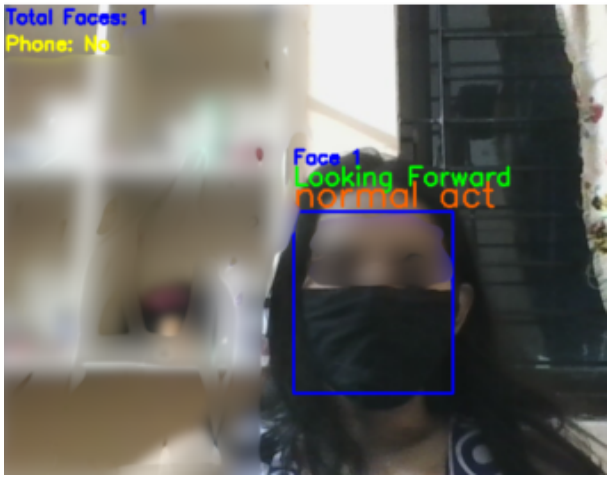

Fig. 12: Also detects masked face and gives correct head pose and CNN output.

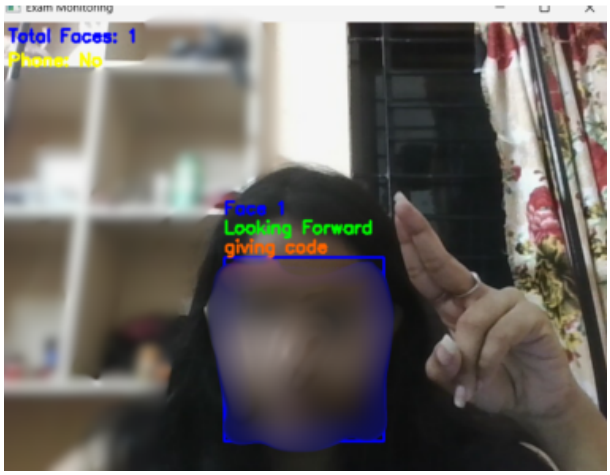

Fig. 13: Showing 'giving code' in the exam and the head looking right.

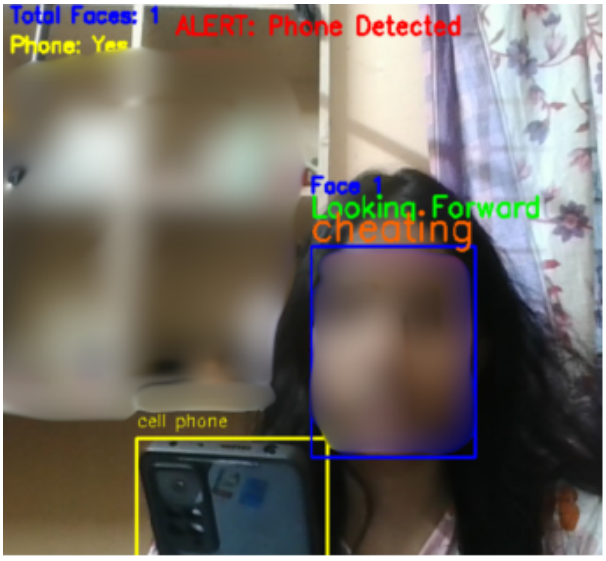

Fig. 14: Showing Alert: Phone Detected and detected cell phone with a yellow bounding box and shows cheating.

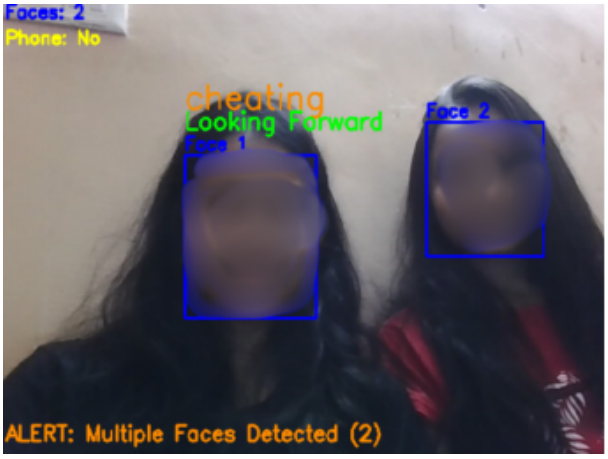

Fig. 15: Showing Alert in case of multiple faces detected while only detecting head pose for one face.

X. IMPACT OF THE PROJECT

This system offers a scalable, automated solution for maintaining fairness in online examinations, reducing the reliance on human invigilators. It helps identify suspicious behaviour such as phone usage or repeated glancing, enhancing exam integrity. By providing real-time alerts and generating logs, it assists institutions in maintaining credible assessment standards. Its modular design makes it adaptable to various environments.

XI. CHALLENGES ENCOUNTERED

1) Maintaining real-time speed with multiple models
2) False positives in phone detection due to hand gestures
3) Consistent head pose across multiple faces
4) Variable lighting conditions

XII. FUTURE SCOPE

1) Audio analysis for whisper detection
2) Deploy as a web/desktop app
3) Improved gaze tracking
4) Re-identification to track specific students
5) Anti-Spoofing Detection.

XIII. CONCLUSION

This project successfully demonstrates a robust real-time malpractice detection system that combines classical computer vision techniques with modern deep learning architectures. By integrating MediaPipe FaceMesh for accurate face tracking and head pose estimation, a custom-trained CNN for behavior classification, and YOLOv8m for mobile phone detection, the system offers a multi-layered approach to monitoring student behavior during online examinations. The system operates on live webcam input, offering real-time feedback and logging to ensure transparency and accountability. With an intuitive overlay that displays head pose direction, predicted action, and device detection alerts, invigilators are equipped with a tool that helps maintain exam integrity at scale. Model training and evaluation metrics validate the effectiveness of the CNN classifier, while the seamless integration with YOLOv8m and MediaPipe enhances detection reliability. Together, these technologies reduce the chances of false positives and increase system robustness across varied conditions such as lighting, angle changes, or multiple faces.

XIV. ACKNOWLEDGMENT

This work was carried out under the guidance of Prof. Anirban Dasgupta and the supervision of Dhruvika Verma at the Department of Electronics and Electrical Engineering, Indian Institute of Technology (IIT) Guwahati, in pursuit of practical advancements in the field of smart systems.

REFERENCES

[1] Ultralytics, "Ultralytics YOLOv8 Documentation," [Online]. Available: https://docs.ultralytics.com/. [Accessed: Aug. 2025].
[2] Google AI, "MediaPipe FaceMesh Guide," [Online]. Available: https://ai.google.dev/edge/mediapipe/solutions/guide. [Accessed: Aug. 2025].
[3] OpenCV Team, "OpenCV Documentation," [Online]. Available: https://docs.opencv.org/. [Accessed: Aug. 2025].
[4] PyTorch Team, "PyTorch Documentation," [Online]. Available: https://pytorch.org/docs. [Accessed: Aug. 2025].
[5] PyTorch Tutorials, "Data Loading and Processing Tutorial," [Online]. Available: https://pytorch.org/tutorials/beginner/data$_{l}oading_{t}utorial.html.[Accessed: Aug.2025].
[6] S. Sahdev, "Head Pose Estimation using OpenCV and Dlib," LearnOpenCV, [Online]. Available: https://www.learnopencv.com/head-pose-estimation-using-opencv-and-dlib. [Accessed: Aug. 2025].
[7] N. Posham *et al.*, "Malpractice Detection with Deep Learning," in *Proc. IEEE Int. Conf. Innovative Computing and Informatics (ICICI)*, 2024, pp. 286–291. Available: https://www.computer.org/csdl/proceedings-article/icici/2024/732900a286/20l58sMeNI4
[8] A. Sharma and R. Patel, "Machine Learning-Based Malpractice Detection in Examinations," SSRN Electronic Journal, 2023. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract$_{i}d$ = 4523598.[$Accessed$: $Aug.$2025].