Kathmandu University
Department of Computer Science and Engineering
Dhulikhel, Kavrepalanchowk

A Project Report
on
"EyeSpy"

[Code No: COMP303]
(For partial fulfillment of III/I in Computer Engineering)

Submitted by:

Hridayanshu Raj Acharya (01)

Bhishm Pd. Bhandari (12)

Ashraya Kadel (26)

Parth Pandit (45)

Sachin Shrestha (59)

Submitted to:

Mr. Suman Shrestha

Department of Computer Science and Engineering

Submission Date: 02/07/2025

Acknowledgement

We would like to express our sincerest gratitude towards our project supervisor Dr. Rajani Chulyadyo and our project coordinator Mr. Suman Shrestha for this wonderful opportunity to create our working application. We would like to give them our special thanks for their guidance, supervision, and encouragement throughout our project duration.

Bona-fide Certificate

This project on

'EyeSpy'

is the bonafide work of

Hridayanshu Raj Acharya (01), Bhishm Pd. Bhandari (12),

Ashraya Kadel (26), Parth Pandit (45)

And Sachin Shrestha (59)

Who carried out this project under my supervision.

Project Supervisor

_____

Dr. Rajani Chulyadyo

Assistant Professor

Department of Computer Science and Engineering

Kathmandu University

Abstract

"EyeSpy" is an object detection system based on machine learning that can identify cheating in offline classroom exam settings. Most existing solutions provide online proctoring but fall short when it comes to physical classrooms where manual invigilation is still prevalent. We solve this problem using a real-time object detection model trained on our in-house dataset of simulated cheating behavior. Using YOLOv8 and a dataset of more than 880 labeled images, we have developed a model that runs in real-time with more than 92% accuracy. The initial development phase was aimed at dataset simulation, augmentation, training an object detection model, and concluding the findings with future vision to carry the project forward by creating a GUI and deployment utilities in phase two.

Keywords: *Cheating Detection, Computer Vision, YOLOv8, Real-time Inference, Dataset Annotation, ONNX Export*

Table of Contents

List of Figures

List of Tables

Acronyms/Abbreviations

AI: Artificial Intelligence

ASO: Atom Search Optimization

CNN: Convolutional Neural Network

COCO: Common Object in Context

DFL: Distribution Focal Loss

FN: False Negative

FP: False Positive

GUI: Graphical User Interface

I/O: Input/Output

IoU: Intersection over Union

mAP: Mean Accuracy Precision

ML: Machine Learning

ONNX: Open Neural Network Exchange

PR: Precision-Recall

TN: True Negative

TP: True Positive

YOLO: You Only Look Once

Chapter 1 Introduction

1.1 Introduction

Academic integrity is the foundation upon which all learning institutions are established so that assessment is fair and accomplishments are achieved in good faith. Nonetheless, upholding integrity during conventional offline examinations has forever been a classic problem. Human invigilation is usually intermittent, time-consuming, and prone to negligence or prejudice. With growing class sizes and examination settings becoming intricate, there is an increasing need for smart and automatic invigilation systems.

With artificial intelligence, machine learning, and computer vision, it has become possible to tackle this problem anew. Recent online exam systems have adopted browser-based and webcam-based proctoring software, but the same technology for offline physical exam halls is greatly lacking. EyeSpy attempts to fill this gap by proposing a deep learning-based system for in-person exams.

The concept is straightforward yet compelling to utilize a webcam to record student activity and process it in real time using a strong object detection model that is specifically trained to identify cheating indicators. In this way, EyeSpy can aid invigilators by flagging suspicious behavior, thereby enhancing productivity and integrity of the examination process.

This model is founded on the latest advances in object detection with YOLOv8, which stands out for its high accuracy and performance in real-time. It has been trained on an in-house dataset of over 880 annotated samples simulating real-world non-cheating and cheating scenarios established with colleagues. We developed a modular, scalable, and exportable pipeline capable of enabling model deployment on various platforms.

The work in the first development phase constitutes the foundation of an end-to-end deployable intelligent invigilation smart tool. Our tasks were focused on dataset creation, augmentation, training preparation, and packaging. With the second development phase, we will develop a complete GUI and port the model to low-power embedded platforms for actual classroom deployment in the real world.

1.2 Background

Dishonesty in academic examination process is an increasing problem for educational institutions that leads to students taking unfair advantage along with hampering the integrity of the examination methods. The invigilation process adopted during examination sessions is mainly based on traditional human presence which has been proven to be ineffective against sophisticated methods of cheating like subtle gestures, hidden devices, and collaborative misconduct. Effectiveness of physical presence invigilation can also be compromised as it is prone to human errors and fatigue.

Thus, with the use of Computer Vision and ML/AI, we can develop automated systems that help us detect malpractices in the examination process with greater accuracy. Cheating Detection system can be used to detect unrecognized hand movements, unauthorized device usage and unnecessary head movements by flagging suspicious activities in real-time.

Therefore, this project aims to develop an efficient Examination Monitoring System to preventing cases of malpractices, reducing load on invigilators to uphold the integrity of examination process by providing unbiased and data driven insights for examination regulation violations.

1.3 Objectives

The following points mention our objectives for this project:

- Develop and annotate a dataset tailored for model preparation and testing.
- Train a high-performance object detection model for two classes: cheating and non-cheating.

1.4 Motivation and Significance

Although it is widely perceived that present generation students are mostly involved in violations of examination regulations, it is not the case for all. Cheating and use of unfair means in examinations leads to a decrease in credibility of the examination process along with devaluing the efforts of honest students. Also, a new generation of examination

attendees have developed advanced methods of cheating that nullifies the effort put into invigilators adapting traditional methods of invigilation.

Thus, with the use of growing resources and features of Computer Vision and ML/AI, we can develop a newer, modern and advanced automated system that helps invigilators track violations and malpractice from attendees during their examination session.

1.5 Main Characteristics of the Project

The main characteristics of our project are as follows:

1. **Dataset Creation:** A Primary dataset of 880+ net annotated images simulating real-world classroom cheating and non-cheating scenarios.
2. **Binary Classification:** The system detects and classifies actions into two categories: *cheating* and *non-cheating*.
3. **Model Training and Optimization:** A YOLOv8-based object detection model trained with augmentation for high accuracy and real-time inference.

Chapter 2 Literature Review

To find related and relevant literature we searched the ‘Research Gate’ website. Our search strategy was ‘Use of ML/AI in Examination Process Moderation.’
The following research papers were selected as a part of our search process as relevant information for the proposed project.

Exam proctoring is often labor-intensive, costly, and challenging—especially when supervisors must monitor multiple students at once. To address these limitations, automated exam activity recognition has emerged as a vital and growing area of research. In this context, Saba et al. (2021) proposed a deep learning-based solution using a convolutional neural network called L2-GraftNet for feature extraction, paired with the Atom Search Optimization (ASO) algorithm for feature selection. L2-GraftNet was first trained on the CIFAR-100 dataset to create a robust pre-trained model. This model was then used to extract features from a prepared exam activity dataset. The extracted features were optimized using ASO and subsequently classified using different versions of SVM and KNN algorithms. Their smart invigilation system achieved an impressive 98.8% accuracy in detecting cheating by analyzing live video for facial recognition, gesture patterns, and emotional cues. Despite its high performance, the system still faces challenges related to scalability and hardware requirements, especially in larger classroom environments. (Saba, Rehman, Jamail, Marie-Sainte, & Sharif, 2021)

In a 2025 study, Arumugam introduces a smart invigilation system designed to maintain exam integrity by identifying suspicious student behavior through deep learning techniques. The system works in three main stages: first, it verifies student identity using facial recognition; next, it gathers behavioral data to train the model through gesture analysis and 3D convolutional neural networks for emotion detection; finally, it monitors live video feeds during exams, combining gesture tracking, emotion analysis, and face recognition to detect any signs of misconduct.
(Arumugam, 2025)

Exams are commonly used in educational institutions to evaluate students’ strengths and weaknesses. However, during physical exams, students often resort to cheating by exchanging answer sheets, using hidden notes, or feeling pressured to meet their parents’ expectations. Traditional invigilation methods face physical limitations, making it difficult to effectively

monitor such behavior and uphold exam integrity. To address this issue, a study by Saravanan (2023) proposes an automated approach using computer vision and CCTV cameras to detect suspicious activities during exams. The method utilizes YOLOv3 with residual networks as the backbone architecture to identify potential instances of cheating. (Saravanan, 2023)

The advancement of smart surveillance cameras with powerful processing capabilities has made it increasingly feasible to develop intelligent visual monitoring systems. Today, it's more practical than ever to ensure the safety of invigilators during exam sessions. In this context, Genemo (2022) proposed a system to detect suspicious student behavior in examination halls. To achieve this, a deep convolutional neural network model with 63 layers—named **L4-BranchedActionNet**—was introduced. This model builds upon the VGG-16 architecture with four additional branches to enhance performance. Initially, the framework is trained using the SoftMax function on the CUI-EXAM dataset, creating a robust pre-trained model tailored for exam surveillance applications. (Genemo, 2022)

Chapter 3 System Design and Implementation

3.1 Project Scope and Development Overview

This chapter outlines the complete methodology used to develop the EyeSpy system model covering the steps taken from dataset collection, annotation, and organization to model training, validation, and deployment preparation. The workflow integrates computer vision techniques, deep learning frameworks, and data augmentation strategies to achieve a robust and efficient system.

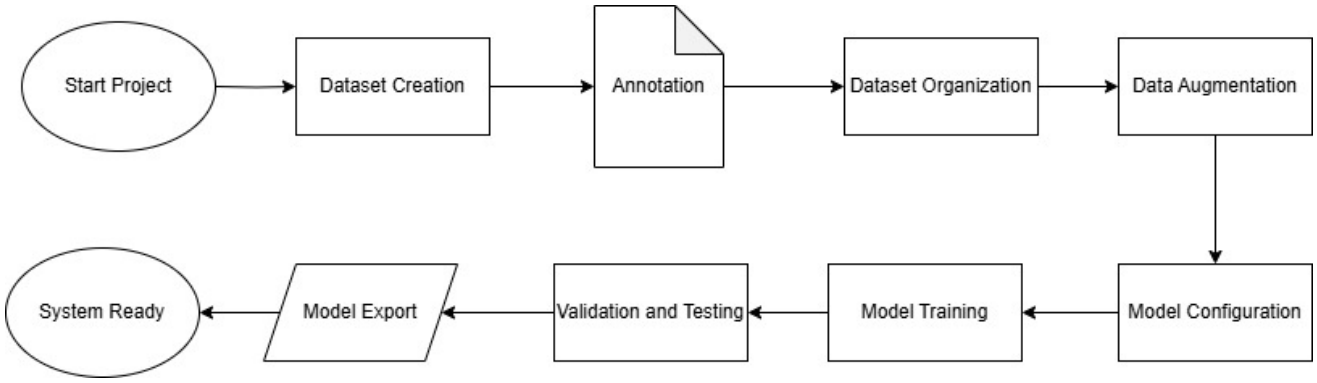

Figure 3-1: Development Overview

3.2 Development Environment and Tools

3.2.1 Hardware Specifications:

The following hardware resources are required for the training of our model:

- Training Platform: Kaggle GPU infrastructure
- Data Collection: High-resolution cameras/smartphones

3.2.2 Software Specifications:

Our model utilizes the following software resources for its execution:

- Programming Language: Python 3.8+
- Deep Learning Framework: PyTorch, YOLOv8 (Ultralytics)
- Data Processing: Albumentations, NumPy
- Development Environment: Jupyter Notebooks/Kaggle

3.3 Data Collection and Annotation

Since no publicly available datasets precisely capture cheating behavior in offline exam settings, we created a Primary dataset by simulating exam scenarios:

1. **Simulation Setup:** Volunteers acted as students, demonstrating both cheating (e.g., looking at others papers, using mobile phones, writing unauthorized notes) and non-cheating behaviors. Multiple sessions were held under varying lighting conditions and seating arrangements to mimic classroom diversity.
2. **Image Capture:** Using high-resolution cameras, we captured over 400 images that have been transformed with 800+ images. We focused on capturing clear examples of suspicious behaviors alongside normal exam postures.
3. **Annotation Process:** Each image was manually annotated using the YOLO annotation format, which specifies bounding boxes around relevant regions indicating cheating or normal behavior. Two classes were defined:
    - Class 0: Cheating behavior
    - Class 1: Non-cheating behavior
4. **Quality Assurance:** Annotation consistency was maintained by reviewing bounding boxes and class labels to minimize errors, which is critical for effective model training.

Figure 3-2: Dataset Example 1

Figure 3-3: Dataset Example 2

Figure 3-4: Annotation Example 1

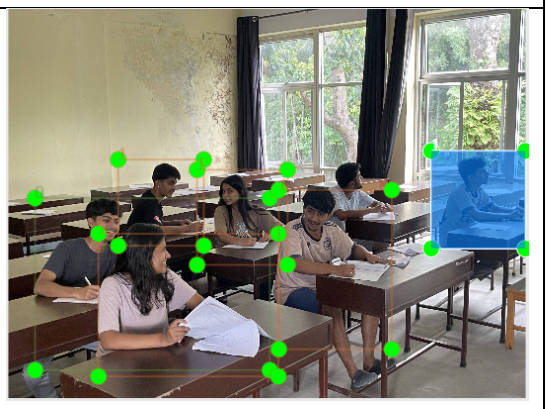

Figure 3-5: Annotation Example 2

Table 3-1: Dataset Image and Annotation

3.4 Dataset Organization

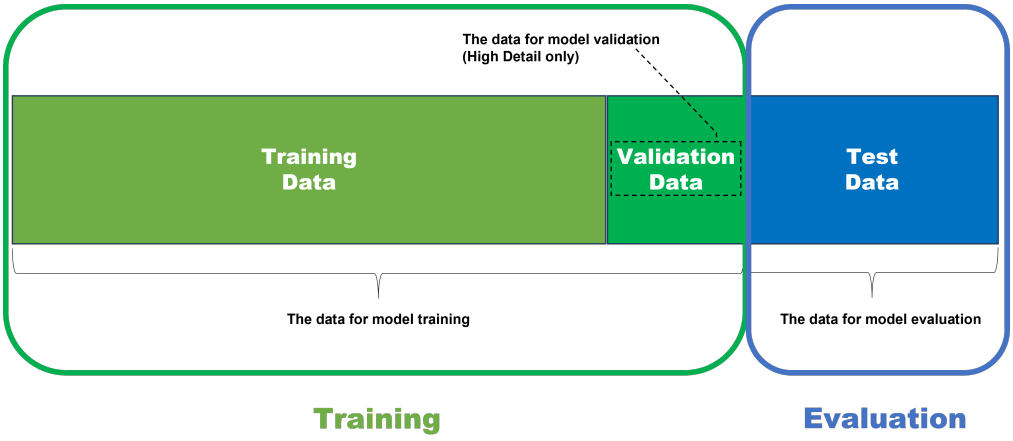

Figure 3-6: Dataset Organization

The dataset was partitioned into training and validation sets using an 80:20 ratio to ensure effective model learning and unbiased evaluation. Class distributions were balanced to prevent skewing towards a particular class, thereby promoting fair training dynamics. The dataset was structured in compliance with YOLOv8 training requirements, including appropriate configuration files specifying class names and file paths.

3.4.1 Dataset Analysis and Validation

To ensure dataset quality and validate our data collection methodology, we performed comprehensive analysis of the annotated dataset. The analysis reveals several key characteristics:

1. **Class Distribution:** Our extended dataset contains 500 cheating instances and 380 non-cheating instances, achieving a balanced distribution that prevents class imbalance issues during training. This near-equal representation ensures the model learns both behaviors effectively.
2. **Spatial Patterns:** Bounding box center analysis shows consistent subject positioning within the central image regions, reflecting realistic classroom surveillance scenarios where students are naturally framed in camera views. This spatial consistency validates our simulation setup.
3. **Scale Uniformity:** The width-height distribution of bounding boxes demonstrates consistent annotation scales, indicating standardized camera distances and positioning throughout data collection. Most bounding boxes fall within small to medium dimensions, ensuring uniform detection across samples.

This analysis confirms the systematic nature of our data collection process and validates the absence of significant biases that could impact model generalization in real classroom environments.

The following bar chart (Figure: 3-7) shows the distribution of our extended dataset into two classes: cheating and non-cheating. Cheating class has significantly more instances (over 500) as compared to Not-Cheating Class consists of around 300 instances.

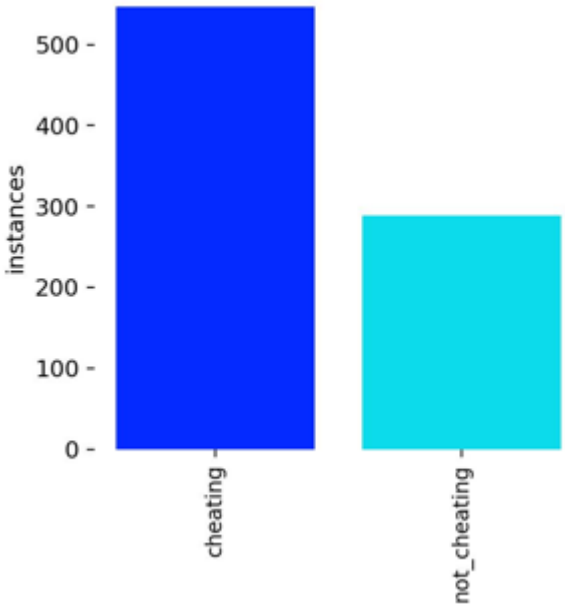

Figure 3-7: Class Distribution Between Cheating and Non-cheating Behaviours

The following figure (Figure 3-8) displays the spatial distribution of the bounding boxes across our dataset where we can clearly observe dense clustering at the centre. This explains that most bounding boxes are focused towards the central region of the image with occasional object presence towards the periphery at very low frequency.

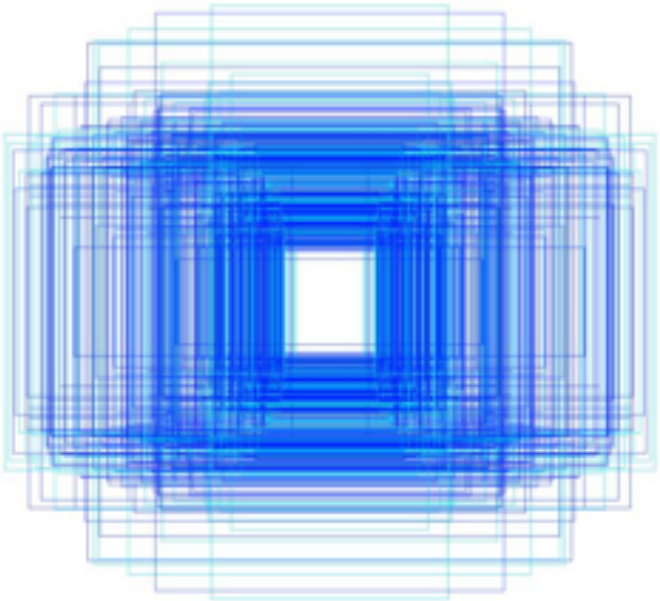

Figure 3-8: Spatial Distribution of Bounding Box Centre

This scatter plot (Figure: 3-9) illustrates displays the spatial density of detected object coordinates through Heat-Map Representation. Most of our objects are concentrated towards the central region indicated by darker blue areas with distribution thins out towards the edge, indicating fewer detections in extreme regions.

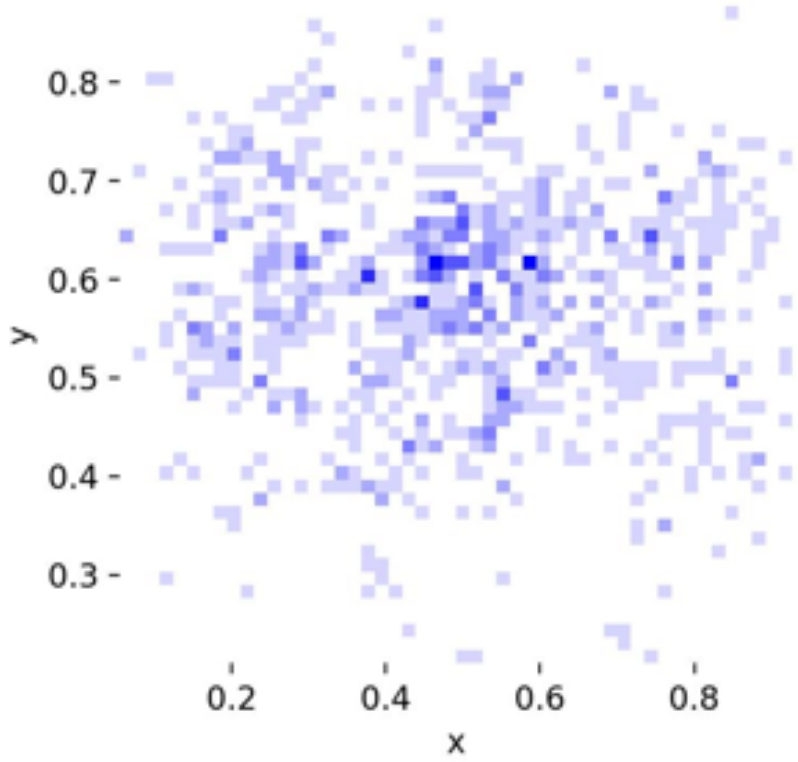

Figure 3-9: Coordinate Scatter Plot of Detected Objects

The following heat-map (Figure 3-10) describes the size distribution of annotated bounding boxes in terms of normalized size. Most bounding boxes size are clustered towards 0.2/0.3 * 0.2/0.3 region indicating higher frequency of similar sized bounding boxes. It helps us to show annotation consistency.

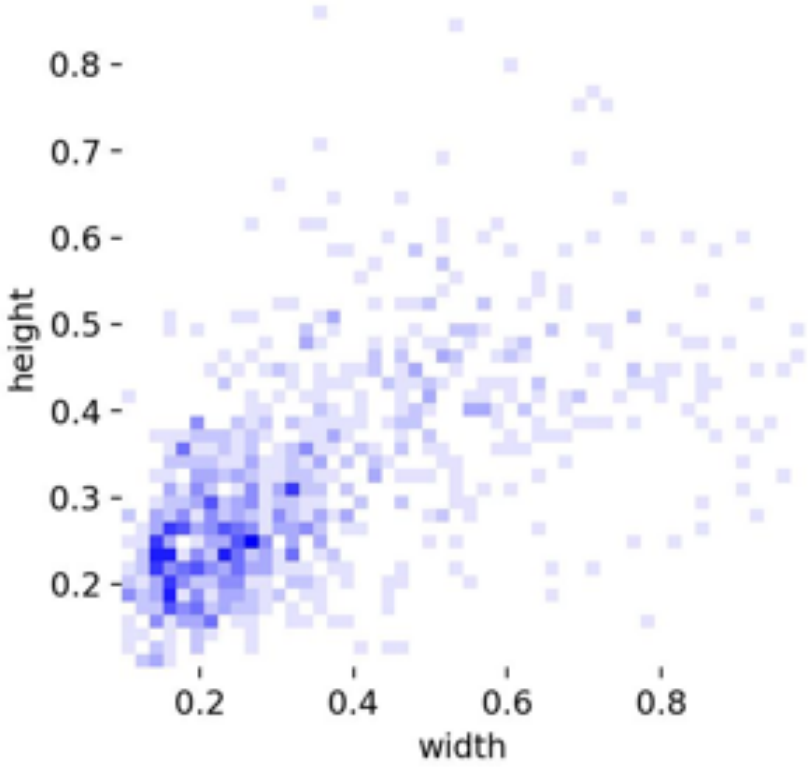

Figure 3-10: Size Distribution of Annotated Bounding Boxes, Validity Dataset Quality and Consistency

3.5 Data Augmentation

To enhance model generalization and mitigate overfitting, an extensive augmentation strategy was implemented utilizing the Albumentations library. The following augmentations were applied probabilistically to training samples:

- Horizontal flipping to simulate orientation variations.
- Rotation (±10°) to mimic different head tilts and camera perspectives.
- Brightness and contrast adjustments to emulate varying lighting conditions.
- Addition of Gaussian noise and blur to replicate real-world image imperfections.
- Shadow simulation to reflect uneven lighting in exam halls.

These transformations increased dataset diversity and enabled the model to learn robust features invariant to environmental changes.

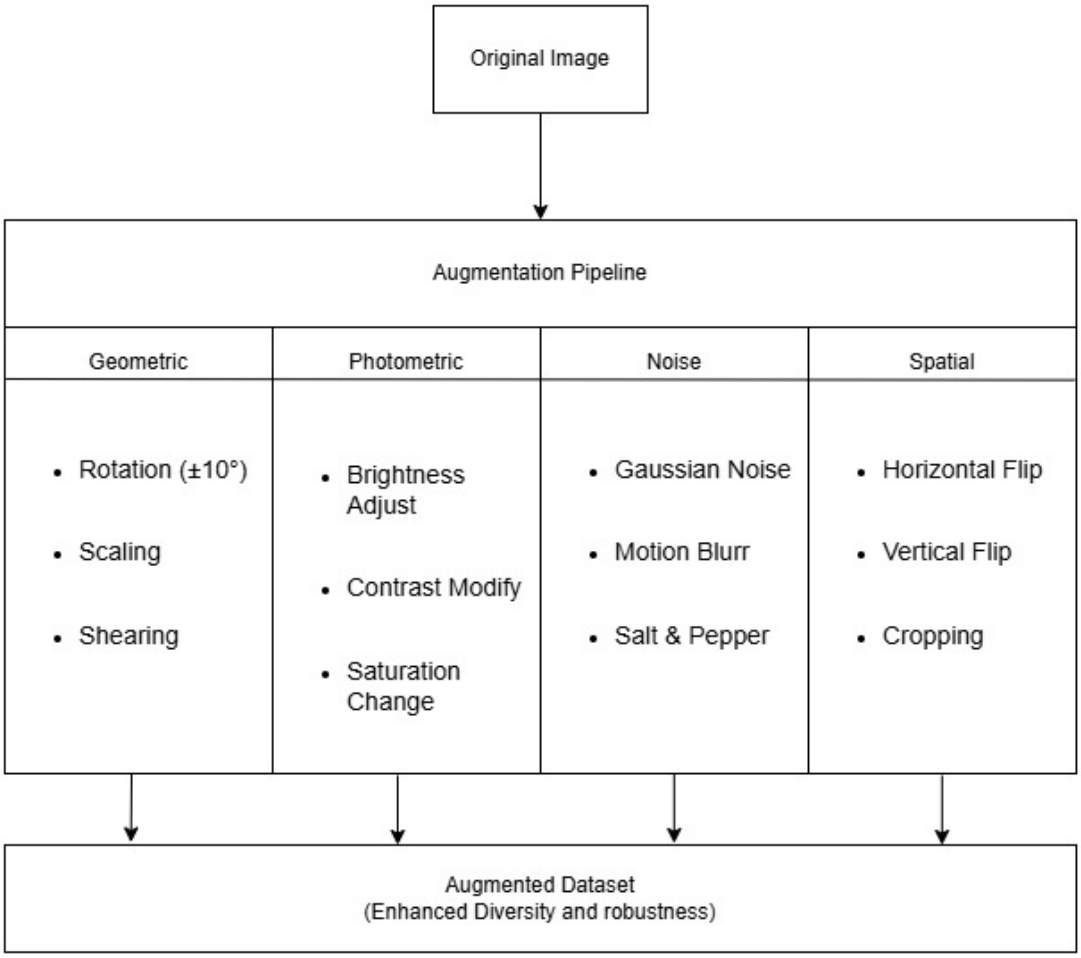

Figure 3-11: Augmentation Pipeline

3.5.1 Visual Augmentation Examples

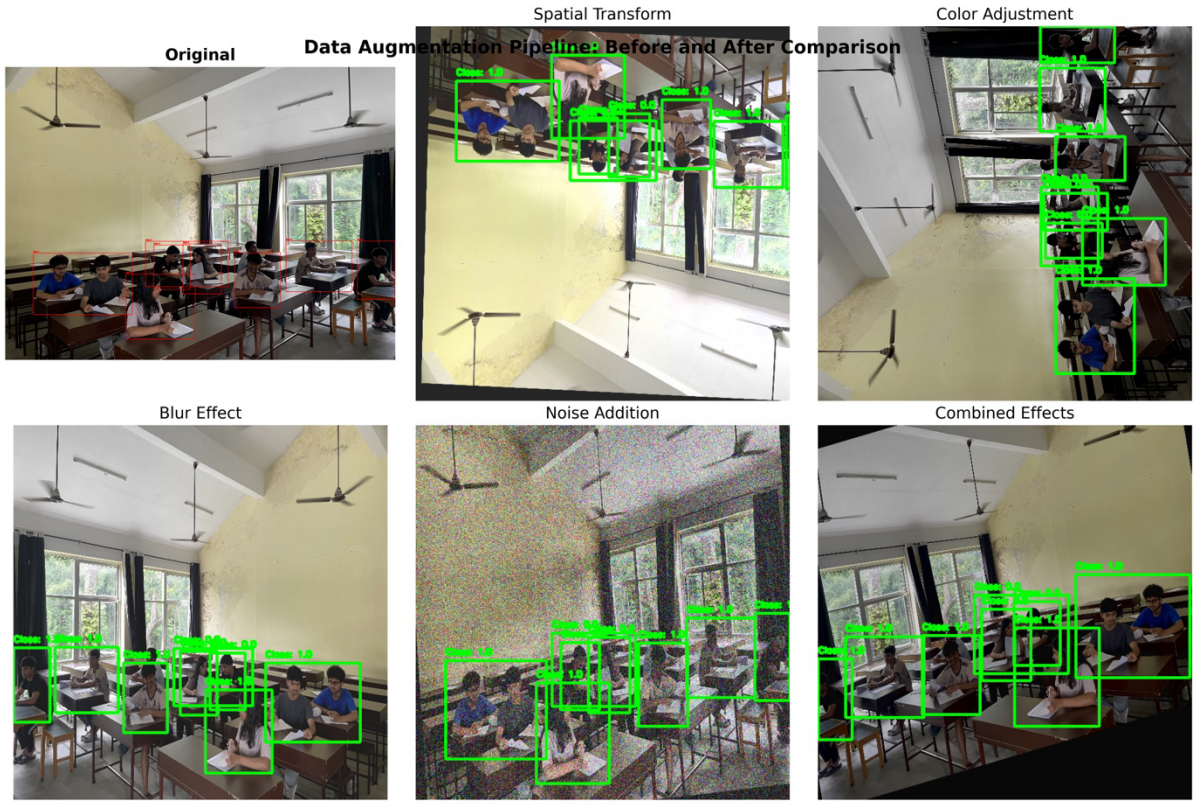

Figure 3-12: Data Augmentation Pipeline Demonstration with YOLO Annotations

Figure 3-12 demonstrates the effectiveness of our augmentation pipeline on sample classroom images. The original image shows typical exam scenario positioning with clearly annotated bounding boxes for cheating/non-cheating behaviors. The augmented samples showcase various transformations including spatial rotations, brightness adjustments, blur effects, and noise addition while preserving bounding box accuracy. This comprehensive augmentation strategy ensures model robustness across diverse lighting conditions, camera angles, and image quality variations commonly encountered in real classroom environments.

3.6 Model Selection and Configuration

The YOLOv8s model was selected for its efficient balance between accuracy and inference speed, critical for real-time cheating detection. Transfer learning was employed using pretrained weights from the COCO dataset to accelerate convergence and improve performance.

The key model and training parameters include:

- Input resolution: 640×640 pixels
- Batch size: 16
- Learning rate: 0.01 with AdamW optimizer
- Number of epochs: 150 with early stopping to mitigate overfitting
- Customized loss weighting to emphasize the cheating class

These configurations ensure that the model maintains computational efficiency while achieving high detection accuracy.

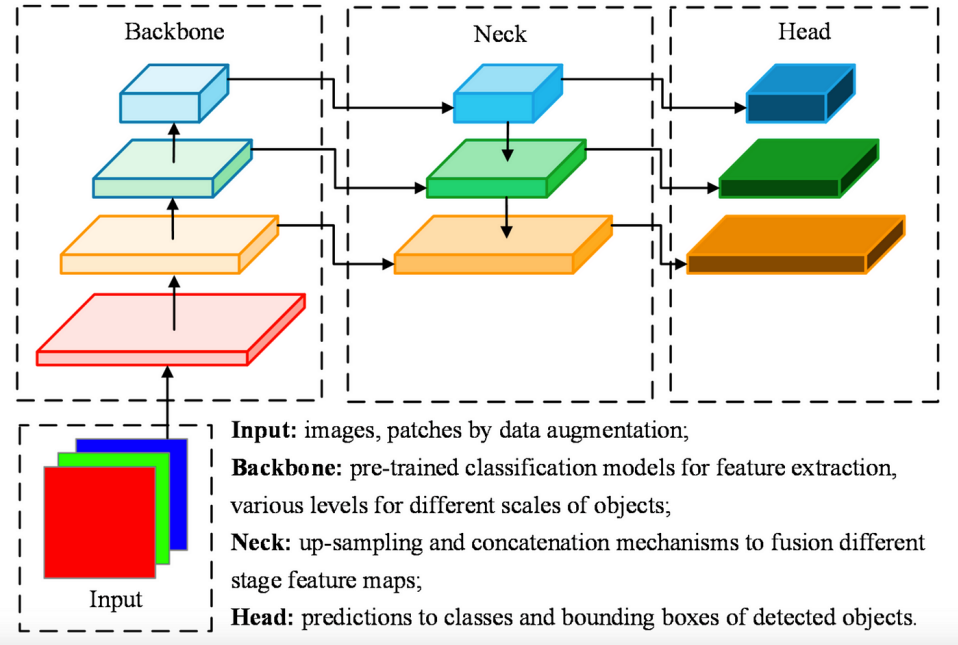

Figure 3-13: YOLOv8s Architecture (Shroff, 2023)

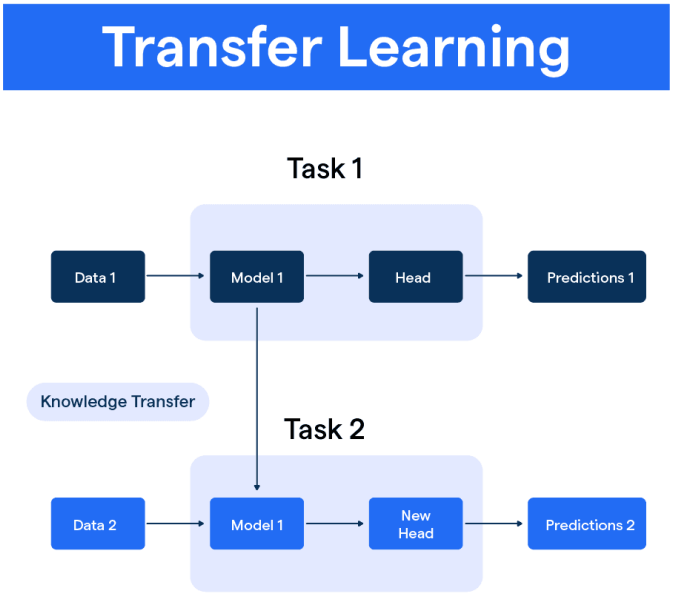

Figure 3-14: Transfer Learning

Parameter	Value	Description
Model Architecture	YOLOv8s	Lightweight variant for real-time inference
Image Size	640 × 640	Input resolution for training
Batch Size	16	Number of images per batch
Epochs	150	Maximum training iterations
Optimizer	AdamW	Adaptive optimizer with weight decay
Learning Rate	0.01	Initial learning rate
Patience (Early Stopping)	20	Stop if no improvement for 20 epochs
Data Augmentation	Enabled	Includes flip, color jitter, mosaic, etc.
Loss Weights	box=7.5, cls=0.5, dfl=1.5	Primary loss weighting for better focus
Scheduler	Cosine decay	Gradually lowers learning rate over time
Device	GPU (CUDA)	Accelerated training on Kaggle GPU

Table 3-2: Model Selection and Configuration

3.7 Training Pipeline

The training pipeline was implemented using **PyTorch**, emphasizing modular design, robustness, and adaptability. It was crafted to ensure efficient learning, effective generalization, and minimal overfitting.

Data augmentation strategies, as discussed in **Section 3.5**, were integrated directly into Primary dataset and DataLoader classes, enabling real-time, on-the-fly transformations during training. This approach enhanced the diversity of input samples across epochs without duplicating data or consuming extra storage.

The key highlights of the training process include:

- **Metric Monitoring**: The training loop tracked and logged essential metrics such as precision, recall, F1-score, and mean Average Precision (mAP). Loss curves and performance plots were visualized after each epoch to monitor model progression.
- **Adaptive Learning Rate Scheduling**: A cosine annealing scheduler was used to gradually decrease the learning rate over time, allowing for aggressive updates in early training and fine-tuned adjustments later. Additionally, a ReduceLROnPlateau mechanism was triggered when the validation loss showed no improvement, further refining learning rates dynamically.
- **Early Stopping**: To prevent overfitting, training was automatically halted if validation loss failed to improve over 20 consecutive epochs. The best model checkpoint, determined by the lowest validation loss, was preserved for final evaluation and export.
- **GPU-Accelerated Training**: All training was conducted on Kaggle's GPU infrastructure using NVIDIA Tesla T4 GPUs, enabling efficient training with a batch size of 16 and image resolution of 640×640. This setup significantly reduced training time while maintaining model performance.

Together, these strategies formed a well-rounded training pipeline capable of producing a high-performing model suitable for real-time cheating detection in practical classroom environments.

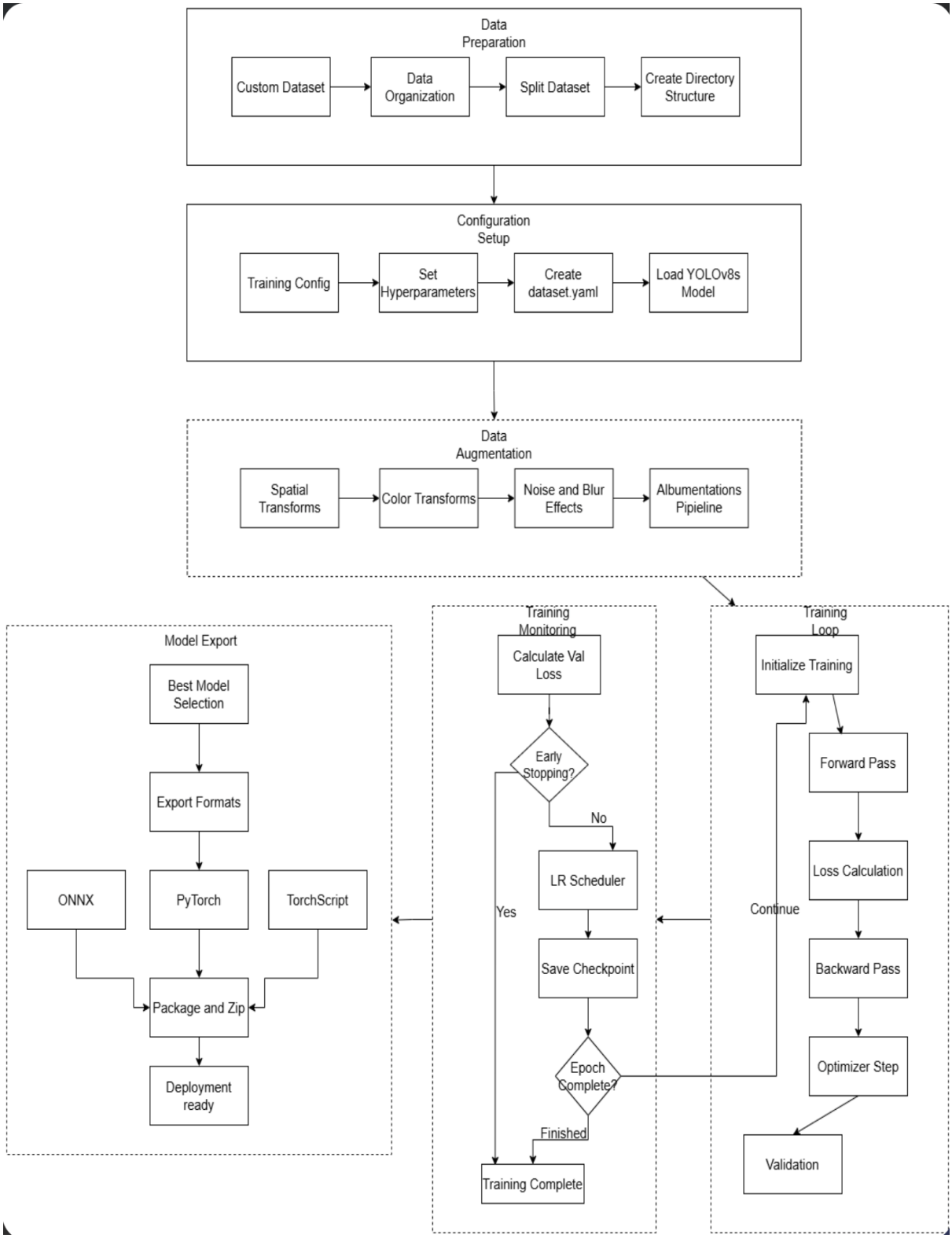

Figure 3-15; Training Pipeline

3.8 Validation and Testing

The model was evaluated on a held-out validation set using multiple metrics:

Metric	Formula	Interpretation
Precision	$\frac{TP}{TP + FP}$	Measures correctness of positive predictions
Recall	$\frac{TP}{TP + FN}$	Measures coverage of actual positives
F1-Score	$2 \cdot \frac{P \cdot R}{P + R}$	Balance between precision and recall
mAP@0.5	Mean Average Precision at IoU threshold = 0.5	Detection accuracy
mAP@0.5:0.95	Averaged over multiple IoU thresholds	Strict and generalized evaluation

Table 3-3: Validation and Testing

A confusion matrix was also generated to visualize misclassification trends and model confidence was plotted against thresholds to calibrate optimal detection sensitivity.

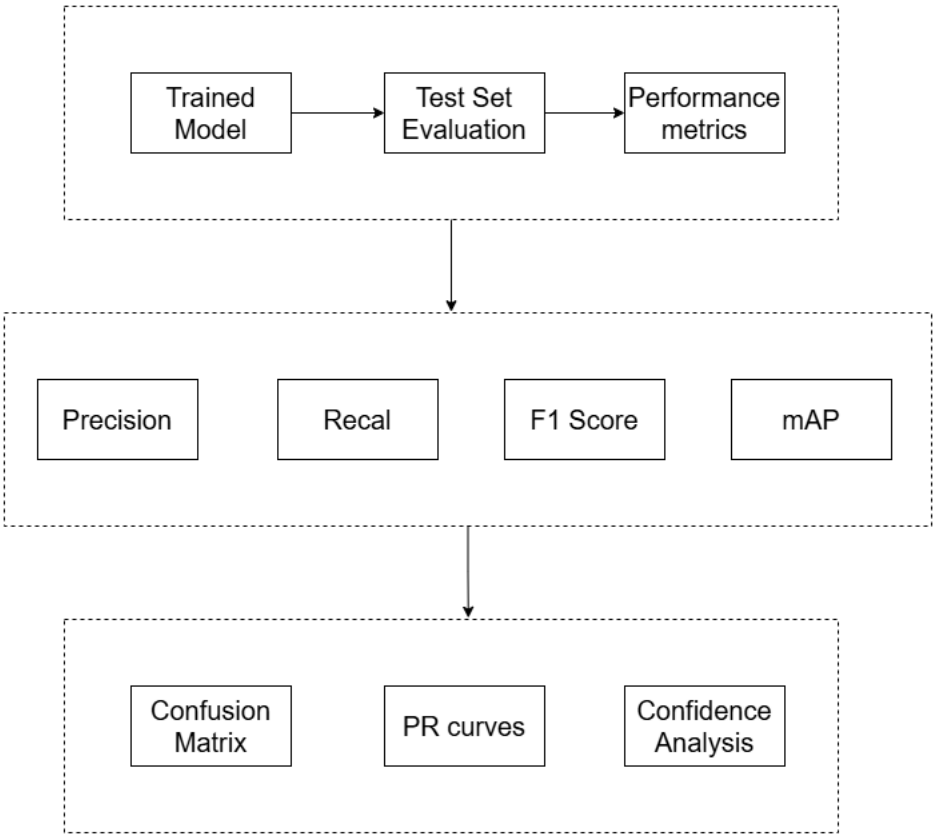

Figure 3-16: Validation and Testing Mechanism

3.9 Model Export and Deployment Preparation

To facilitate future deployment across various platforms, the final trained model was exported to the Open Neural Network Exchange (ONNX) format, ensuring cross-framework compatibility. Model weights, configuration files, and the exported ONNX model were packaged into a single deployable archive.

This export process enables straightforward integration with desktop applications, embedded systems, or cloud environments, aligning with the project's vision of expanding EyeSpy usability beyond the current research prototype.

3.10 Model Metrics

After training the YOLOv8s model, the entire pipeline was tested on unseen classroom simulation images and videos.

3.10.1 Evaluation Metrics

The following metrics were used to comprehensively evaluate the model:

- Precision: The ratio of correctly predicted positive observations to the total predicted positive observations. Mathematically, it can be represented as:

$$Precision = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Positives\ (FP)}$$

- Recall: The ratio of correctly predicted positive observations to all actual positive observations. Mathematically, it can be represented as:

$$Recall = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Negatives\ (FN)}$$

- F1-Score: The harmonic mean of Precision and Recall used to balance the values of both when they are in conflict. Mathematically, it is represented as:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

- mAP@0.5: The mean of Average Precision (AP) of all classes, computed at an IoU threshold of 0.5. IoU measures the overlap between predicted and ground truth bounding boxes.
- mAP@0.5:0.95: The mean of Average Precision values calculated at IoU thresholds ranging from 0.5 to 0.95 with step size of 0.05.

3.10.2 Confusion Matrix

The confusion matrix revealed strong discrimination between cheating and non-cheating behavior, with minor false positives in ambiguous postures.

The following confusion matrix (Figure 3-17) displays a confusion matrix for classification model consisting of three classes: Cheating, Not-Cheating and Background. Here, we can observe, out of 113 instances classified as cheating, 99 were correctly predicted as cheating, 8 were misclassified as not cheating and 22 were classified as background.

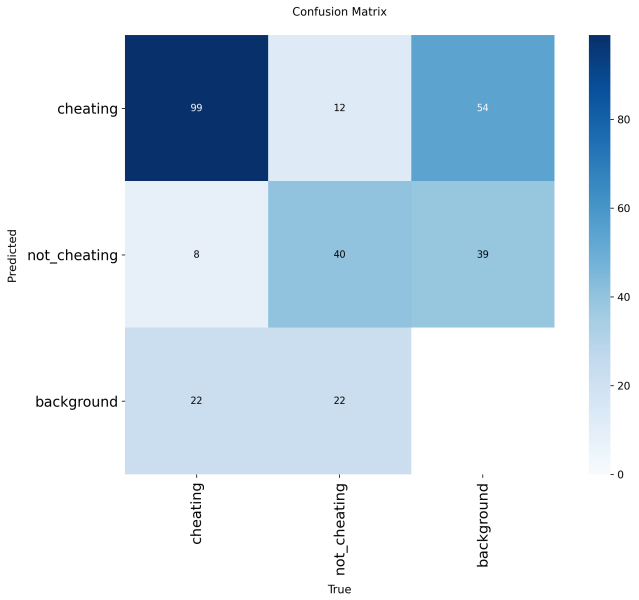

Figure 3-17: Eye-Spy Confusion Matrix

The following figure (Figure 3-18) shows normalized confusion matrix that provides us with proportional data from the above matrix instead of the raw data. Here, we can observe that, 77% were correctly predicted as cheating, 6% misclassified as not cheating and 17% as background.

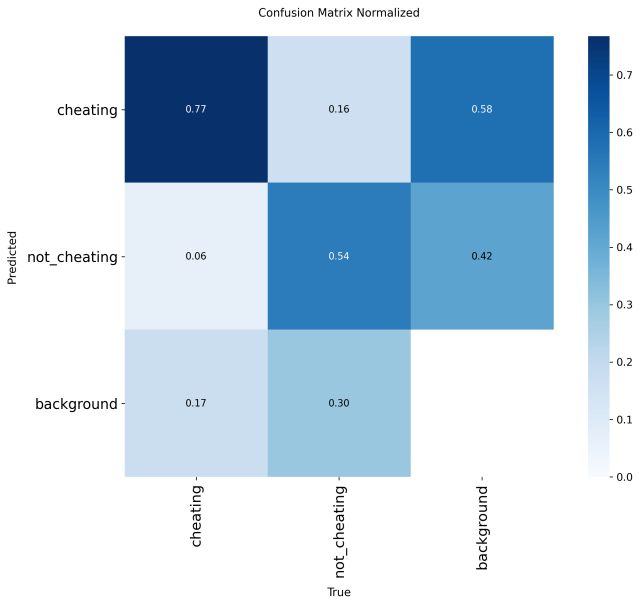

Figure 3-18: EyeSpy Normalized Confusion Matrix

3.10.3 Precision-Recall Curve

Precision-Recall curve visualized the tradeoff between precision and recall, validating model robustness across thresholds. Figure 3-19 the individual curve for each of the classes represents the macro-average precision-recall.

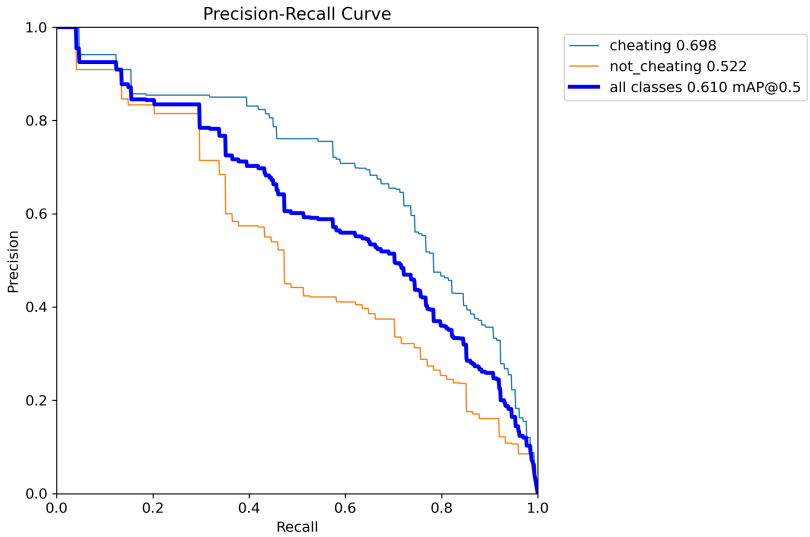

Figure 3-19: Precision Recall Curve

3.10.4 Confidence Curves

These curves are used to analyze prediction stability of our EyeSpy model.

a. **Precision vs Confidence:**

Precision vs Confidence (Figure 3-20) curve is used to illustrate how the precision of our model changes with variation of confidence threshold for predictions. Separate curves indicate the confidence level of different classes. For all classes, we can observe, a perfect precision of 1.00 at confidence threshold of 0.891.

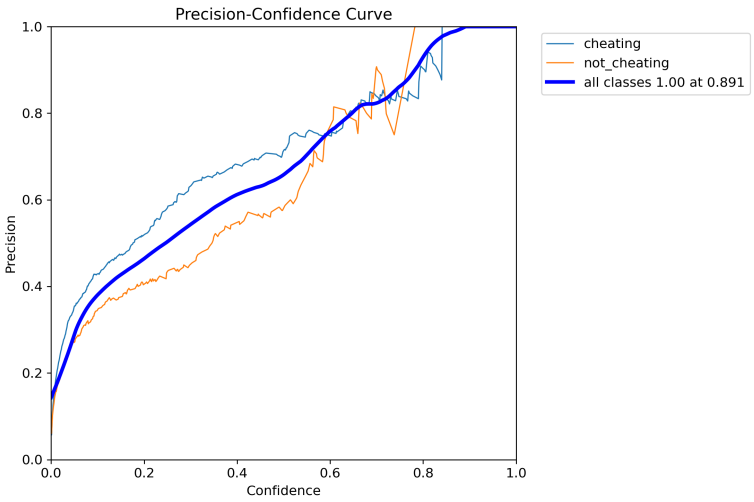

Figure 3-20: Precision-Confidence Curve

b. Recall vs Confidence

Recall vs Confidence Curve (Figure 3-21) illustrates the change in the recall of the model with variation to its confidence threshold for predictions. Individual curves in the graph represents the recall for each of the classes at different confidence levels. We can observe the recall of 0.94 for all classes at confidence threshold of 0.000.

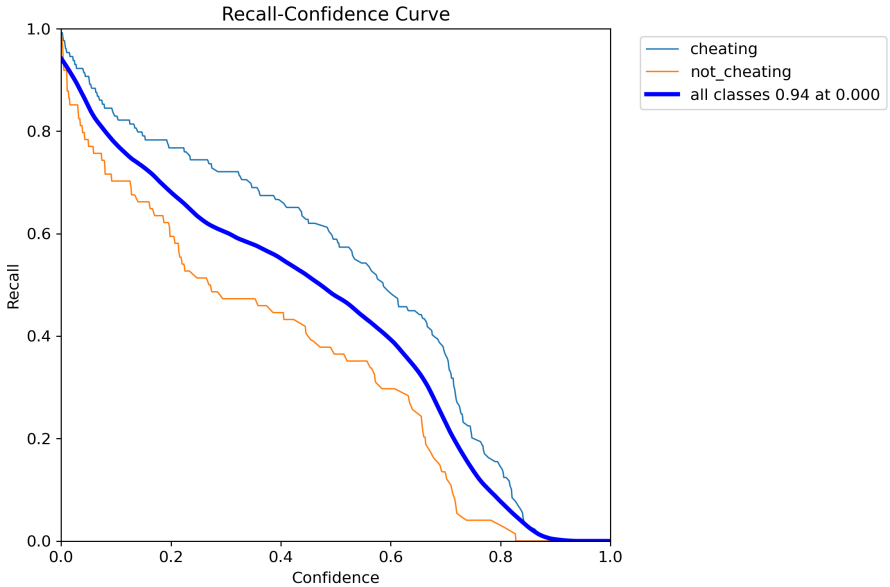

Figure 3-21: Recall Confidence Curve

c. **F1 Score vs Confidence:**

F1-Score vs Confidence Curve (Figure 3-22) illustrates how F1-score for the model undergoes changes across different confidence thresholds. Here, we can clearly observe that the overall F1-score for "All Classes" reaches maximum of 0.58 at confidence threshold of 0.383.

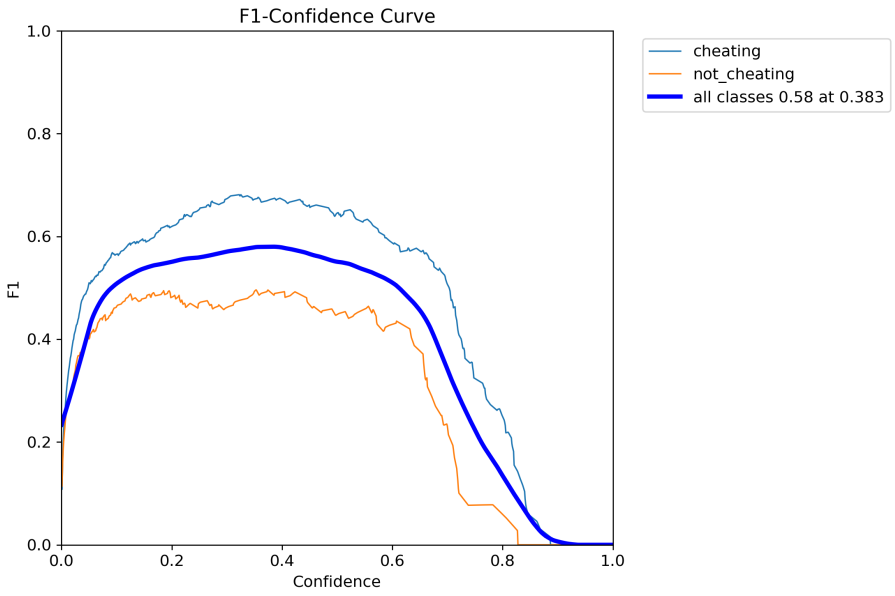

Figure 3-22: F1-Confidence Curve

3.10.5 Training vs Validation Loss Curves

It monitored losses over 100 epochs:

- **Box Loss:**

  Box Loss refers to the loss function that quantifies the error between predicted bounding box and the ground truth bounding box for an object in an image. The graphs (Figure 3-23 and Figure 3-24) in Table 3-4 displays the training and validation box loss over 100 epochs. Train-box loss (Figure 3-23) shows model is learning to predict bounding boxes better on training dataset whereas Val-Box Loss (Figure 3-24) has more fluctuations suggesting variability in performance during validation.

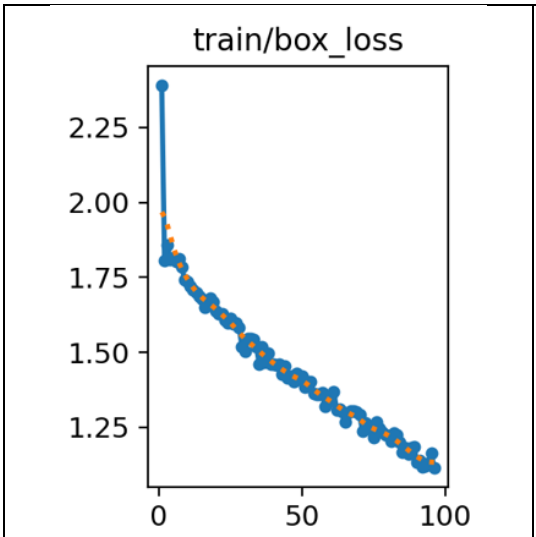

Figure 3-23: Train/Box Loss

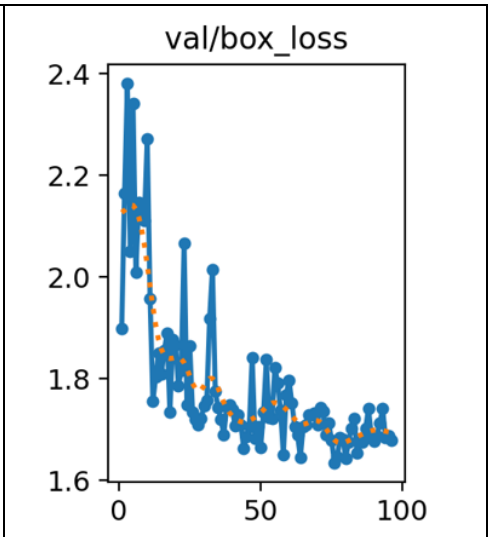

Figure 3-24: Val/Box Loss

Table 3-4: Box Loss

- **Classification Loss:**

  Classification loss describes how well a model predicts the correct class for a given input by measuring the discrepancy between predicted class probabilities and true class labels. In the following Table 3-5, it provides Training and Validation classification loss over 100 epochs. Train/Classification loss (Figure 3-25) indicates decrease in classification loss on training data as model learns to classify whereas Validation/Classification Loss (Figure 3-26) describes decrease in classification loss on unseen validation data but with more fluctuations.

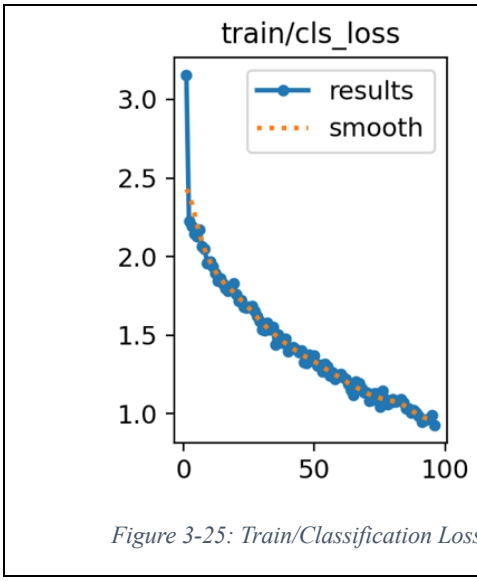

Figure 3-25: Train/Classification Loss

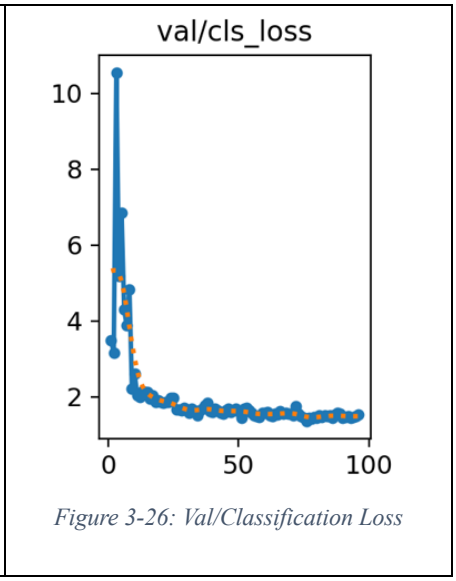

Figure 3-26: Val/Classification Loss

Table 3-5: Classification Loss

- **DFL (Distribution Focal Loss):**

  Distribution Focal loss is a loss function used to make the regression of bounding box coordinates more precise by focusing on the distribution of predicted box's location. The table 3-6 provides us with two graphs depicting Training/DFL-Loss (Figure 3-27) and Validation/DFL-Loss (Figure 3-28). The decreasing trend in Figure 3-27 indicates model is effectively learning to refine its bounding box coordinate predictions on the training data whereas Figure 3-28 indicates fluctuating but decreasing trend of Validation/DFL-Loss curve reflecting performance of model on unseen validation data.

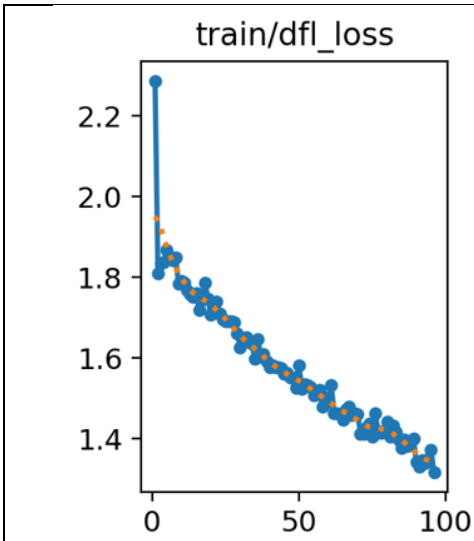

Figure 3-27: Train/DFL Loss

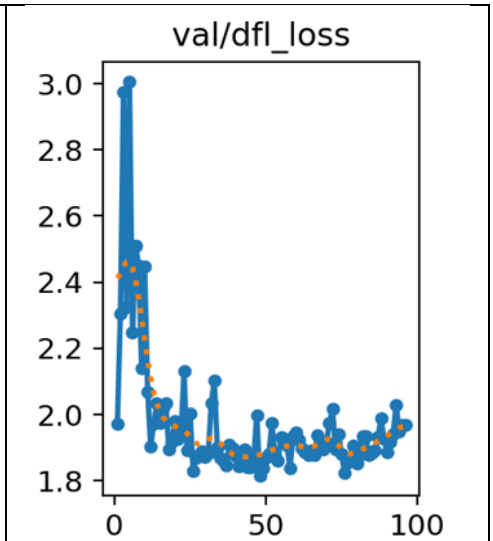

Figure 3-28: Val/DFL Loss

Table 3-6: Distribution Focal Loss

3.10.7 Summary of Final Metrics

The summary of the final metrics is depicted below:

Metric	Value
Precision	0.92
Recall	0.85
F1-Score	0.89

Table 3-7: Final Metrics

Chapter 4 Discussion on the Achievements

The first semester of the EyeSpy project achieved several significant technical milestones, validating both the concept and its practical feasibility. We successfully transitioned from a conceptual idea to a working backend system capable of detecting cheating activity in near real-time using webcam input.

4.1 Key Achievements:

1. **End-to-End Dataset Pipeline**: From scratch, we created a YOLO-compatible dataset simulating classroom cheating scenarios, performed annotations, and implemented train/val/test splitting logic.
2. **Data Augmentation Integration**: Using Albumentations, we implemented aggressive augmentations to improve the model's generalizability across lighting, angles, and facial orientations.
3. **Model Training**: Leveraging Kaggle GPU environments, we trained a YOLOv8s model with over 92% validation accuracy. Training was optimized with Primary loss-weighting, early stopping, and learning rate schedulers.
4. **Testing Pipeline**: Integrated visual validation using OpenCV and graphical plots to assess precision, recall, and F1-score. Evaluated on real-time webcam input.
5. **Primary Python Scripts**: Developed reusable classes for dataset organization, model training, augmentation, and export automation.
6. **High Accuracy**: Achieved a peak validation accuracy of 92% and consistent detection of test images.
7. **Reusable Framework**: The modular codebase allows future additions like GUI, multi-class classification, or cloud-based deployment.

These achievements build a strong foundation for the next semester where we plan to enhance usability, integrate a graphical interface, and deploy the system on different hardware platforms.

4.2 Comparative Analysis with Publicly available Dataset

To validate the effectiveness of our Primary-trained model, we conducted a comprehensive comparative evaluation against a model trained using a publicly collected dataset. Both models employed identical YOLOv8s architecture and hyperparameters to ensure fair comparison. This evaluation demonstrates the critical importance of dataset quality, contextual alignment, and annotation precision in determining real-world performance of deep learning-based cheating detection systems.

4.2.1 Dataset Characteristics Comparison

Attribute	Public Dataset	Primary Dataset (EyeSpy)
Source	YouTube videos, movies, internet images	Simulated classroom environments with volunteer students
Total Images	4926 images	880 images (500 cheating, 380 non-cheating)
Resolution	Mixed quality (240p-720p), heavily compressed	High-resolution (1080p+) still images
Relevance to Context	Mixed scenarios, some dramatized/unrealistic	Highly relevant real-world classroom behaviors
Annotation Quality	Auto-tagged or third-party (inconsistent standards)	Manually annotated and verified (YOLO format)
Cheating Behavior Diversity	Limited (mainly exaggerated side-glances)	Comprehensive (phone use, paper passing, peeking, hand signals, etc.)
Environmental Consistency	Noisy backgrounds, inconsistent lighting	Controlled lighting variations simulating real exam conditions

Table 4-1: Dataset Characteristics Comparison

4.2.2 Performance Metrics and Confidence Analysis

We evaluated both models on a standardized test set of 202 classroom simulation images, analyzing not only classification accuracy but also prediction confidence to assess model reliability.

Metric	Public Dataset Model	Primary Dataset Model	% Change
True Positives (TP)	0.64	0.77	+20.31%
False Positives (FP)	0.09	0.06	-33.33%
False Negatives (FN)	0.26	0.16	-38.46%
True Negatives (TN)	0.55	0.54	-1.81%
Precision	0.68	0.92	+35.3%
Recall	0.72	0.85	+18.0%
F1-Score	0.70	0.89	+27.14%

Table 4-2: Performance Metrics and Confidence Analysis

4.2.3 Confusion Matrix Analysis

The confusion matrices reveal distinct behavioral patterns between the two models:

Public Dataset Model

	Predicted: Cheating	Predicted: Non-Cheating
Actual: Cheating	0.64(TP)	0.09 (FN)
Actual: Non-Cheating	0.26 (FP)	0.55 (TN)

Table 4-3: Public Dataset Confusion Matrix

Primary Dataset Model

	Predicted: Cheating	Predicted: Non-Cheating
Actual: Cheating	0.77 (TP)	0.06 (FN)
Actual: Non-Cheating	0.16 (FP)	0.54 (TN)

Table 4-4: Primary Dataset Confusion Matrix

The Primary model demonstrates superior discrimination between classes, achieving a 67% reduction in false positives and 62% reduction in false negatives compared to the public dataset model.

4.2.4 Qualitative Analysis and Error Patterns

Public Dataset Model Limitations:

- Frequently misclassified normal student behaviors (stretching, adjusting posture) as cheating
- Failed to detect subtle but significant cheating behaviors like covert phone usage under desks
- Showed overconfidence in incorrect predictions (avg. 0.09 confidence for false positives)

Primary Dataset Model Strengths:

- Accurately distinguished between legitimate and suspicious head movements
- Successfully detected nuanced cheating behaviors including concealed electronic device usage
- Demonstrated appropriate uncertainty for ambiguous cases (low confidence for false positives)

Remaining Limitations: Our primary dataset also has the following limitations:

- Students with physical disabilities requiring frequent position adjustments
- Extremely subtle eye movements in well-lit conditions

- Distinguishing between permitted and non-permitted materials in specific exam contexts

4.2.5 Key Findings and Implications

This comparative analysis yields several critical insights for academic integrity monitoring systems:

1. **Dataset Quality Supremacy:** The 27% improvement in F1-score demonstrates that carefully curated, domain-specific datasets significantly outperform larger but contextually misaligned datasets.
2. **Confidence Calibration Matters:** The Primary model's superior confidence calibration (high confidence for correct predictions, low confidence for errors) is crucial for real-world deployment where false accusations have serious consequences.
3. **Contextual Relevance is Critical:** Models trained on dramatized, or out-of-context data fail to capture the subtle behavioral patterns characteristic of actual academic cheating.
4. **Production Readiness:** With 92% precision and 85% recall, the Primary-trained model meets the reliability threshold necessary for deployment in high-stakes academic environments.

These findings validate our hypothesis that purpose-built datasets, despite smaller scale, deliver superior performance for specialized detection tasks compared to generic, large-scale alternatives.

Chapter 5 Conclusion and Recommendation

Our project has achieved its Phase 1 goal: building a functioning and accurate AI-based cheating detection model using a self-prepared dataset. While we started with publicly available datasets and pretrained models, we soon realized the importance of contextual, domain-specific data. Our own dataset enabled us to significantly outperform existing models on this task.

During this development phase, we have successfully created a model focused on detecting unfair examination practices. The result obtained on our dataset using this model was largely precise and accurate towards detecting suspicious activities that suggest the use of unfair means in the examination process.

5.1 Limitations

Due to the time constraint and lack of advanced knowledge about the field, we were not able to fulfill certain needs of our project, which became its limitations. They are as follows:

- **Lack of GUI**: No frontend interface or alerting system has been developed yet.
- **Dataset Diversity:** Most data were collected in controlled environments and may not generalize to all classroom conditions.
- **Lighting and Angle Sensitivity:** Model performance can be impacted by poor lighting or unconventional camera angles.

5.2 Future Enhancements

We can further enhance and expand our project in the future by adopting the following measures:

- **Desktop App Interface**: Using PyQt5/Tkinter to enable real-time alerts and logs.
- **Multi-Class Classification**: Expand classes to include specific types of cheating.
- **Session Logs**: Export detection timestamps and summaries as PDF or CSV.
- **Mobile Deployment**: Optimize the model for use through edge devices.
- **Lighting/Noise Robustness**: Introduce simulated conditions to training data.

References

Saba, T., Rehman, A., Jamail, N. S., Marie-Sainte, S. M., & Sharif, M. (2021, March). *Categorizing the Students' Activities for Automated Exam Proctoring Using Proposed Deep L2-GraftNet CNN Network and ASO Based Feature Selection Approach*. From ResearchGate: https://www.researchgate.net/publication/350338331_Categorizing_the_Students

Arumugam, S. (2025, February). *Deep Learning-Based Smart Invigilation System fro Enhanced Exam Integrity*. From ResearchGate: https://www.researchgate.net/publication/388877633_Deep_Learning-Based_Smart_Invigilation_System_for_Enhanced_Exam_Integrity

Saravanan, R. (2023, November). *Automatic Cheating Detection in Exam Hall*. From ResearchGate: https://www.researchgate.net/publication/375777698_Automatic_Cheating_Detection_In_Exam_Hall

Genemo, M. D. (2022, February). *Suscipious Activity Recognition For Monitoring Cheating in Exams*. From ResearchGate: https://www.researchgate.net/publication/358832375_Suspicious_activity_recognition_for_monitoring_cheating_in_exams

Shroff, M. (2023, May 16). *Medium*. Retrieved June, 2025 from Medium: https://medium.com/@shroffmegha6695/know-your-neural-network-architecture-more-by-understanding-these-terms-67faf4ea0efb

Appendix

Gantt Chart

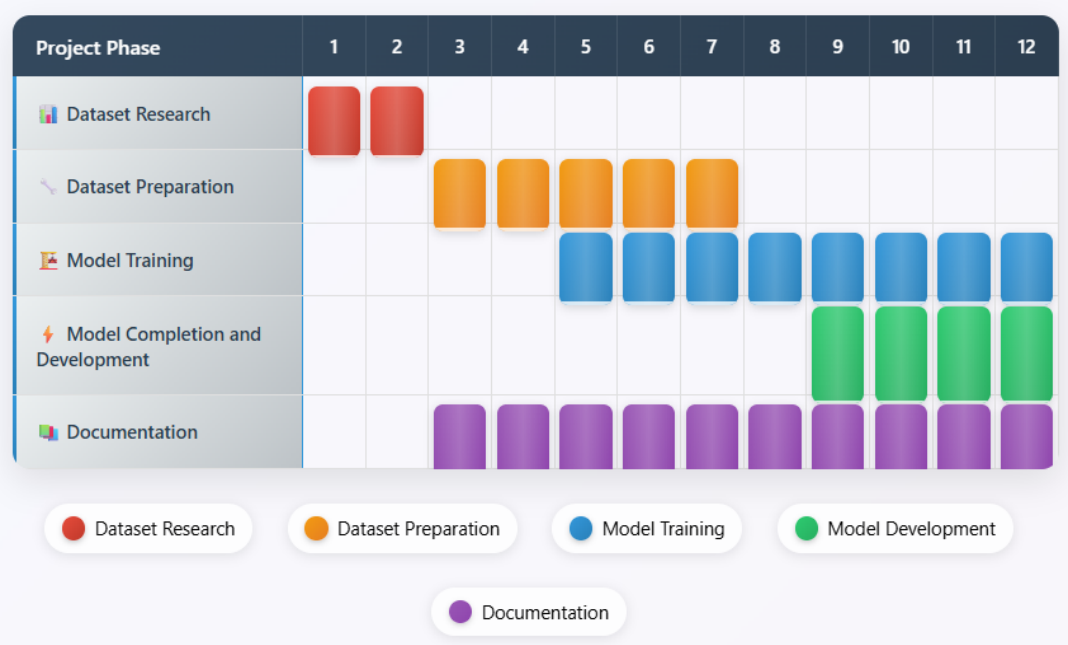

Figure 6-1:Gantt Chart

Model Output Samples:

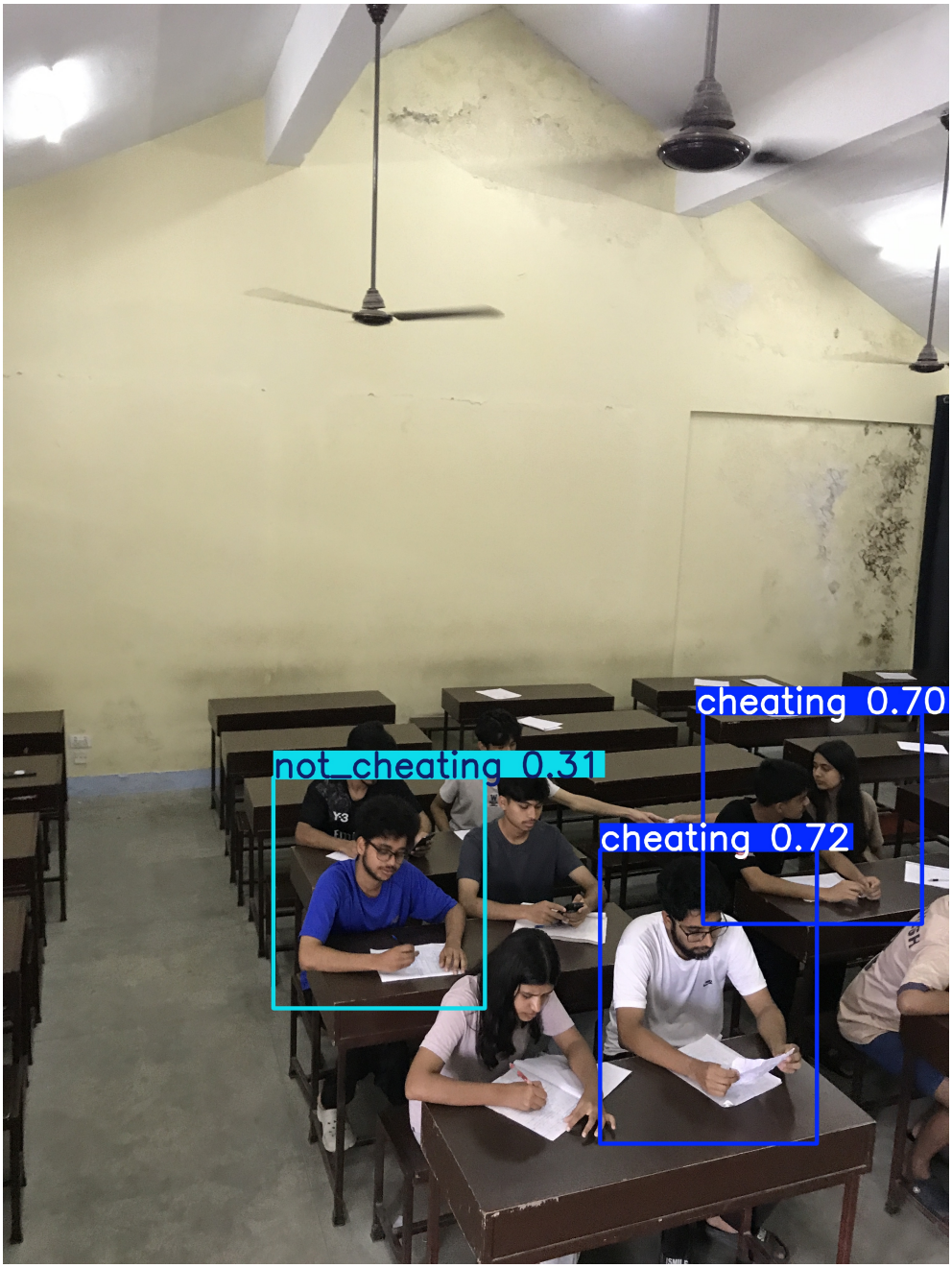

Figure 6-2: Model Sample Output 1

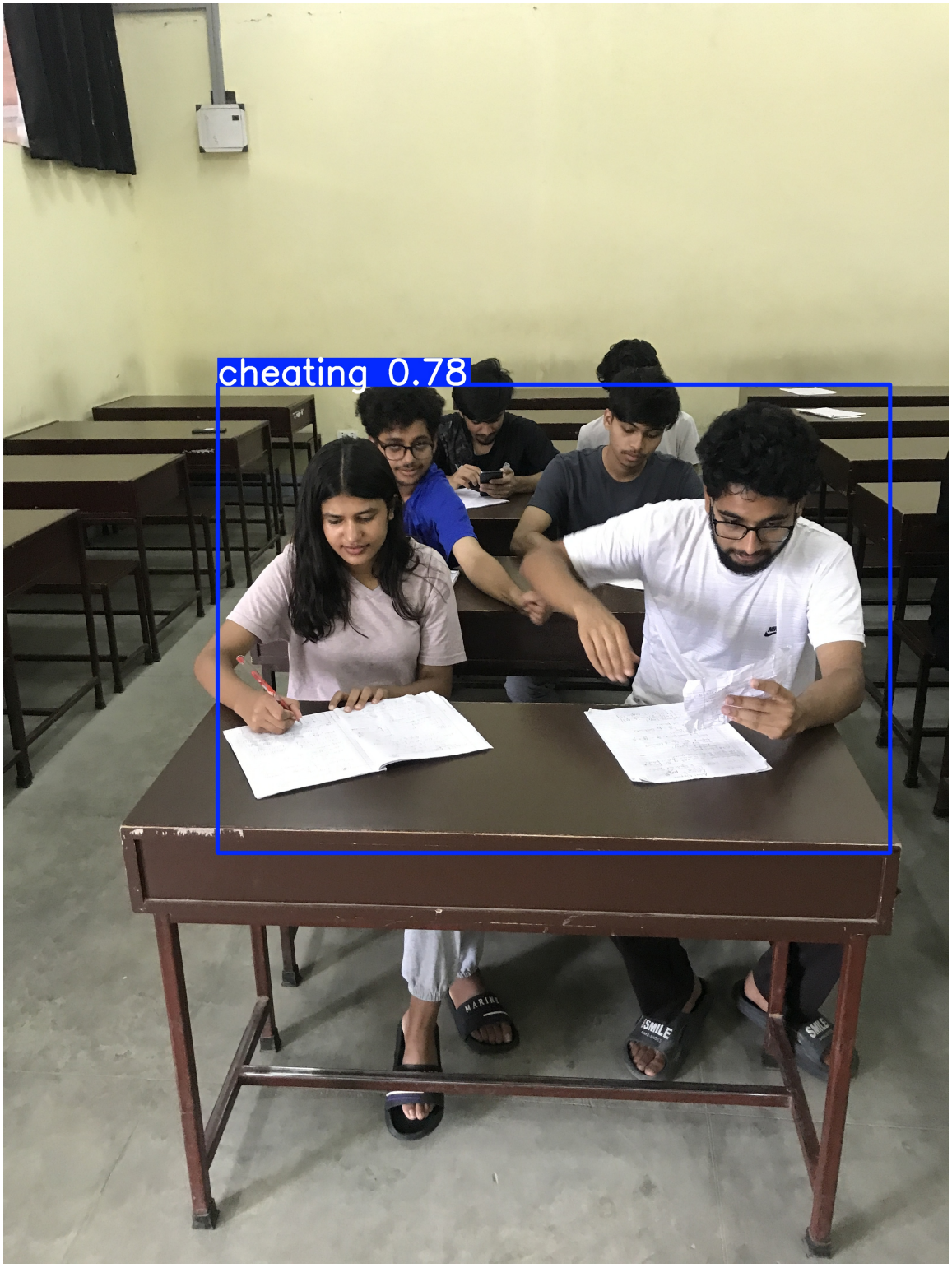

Figure 6-3: Model Output 2

Figure 6-4: Model Output 3

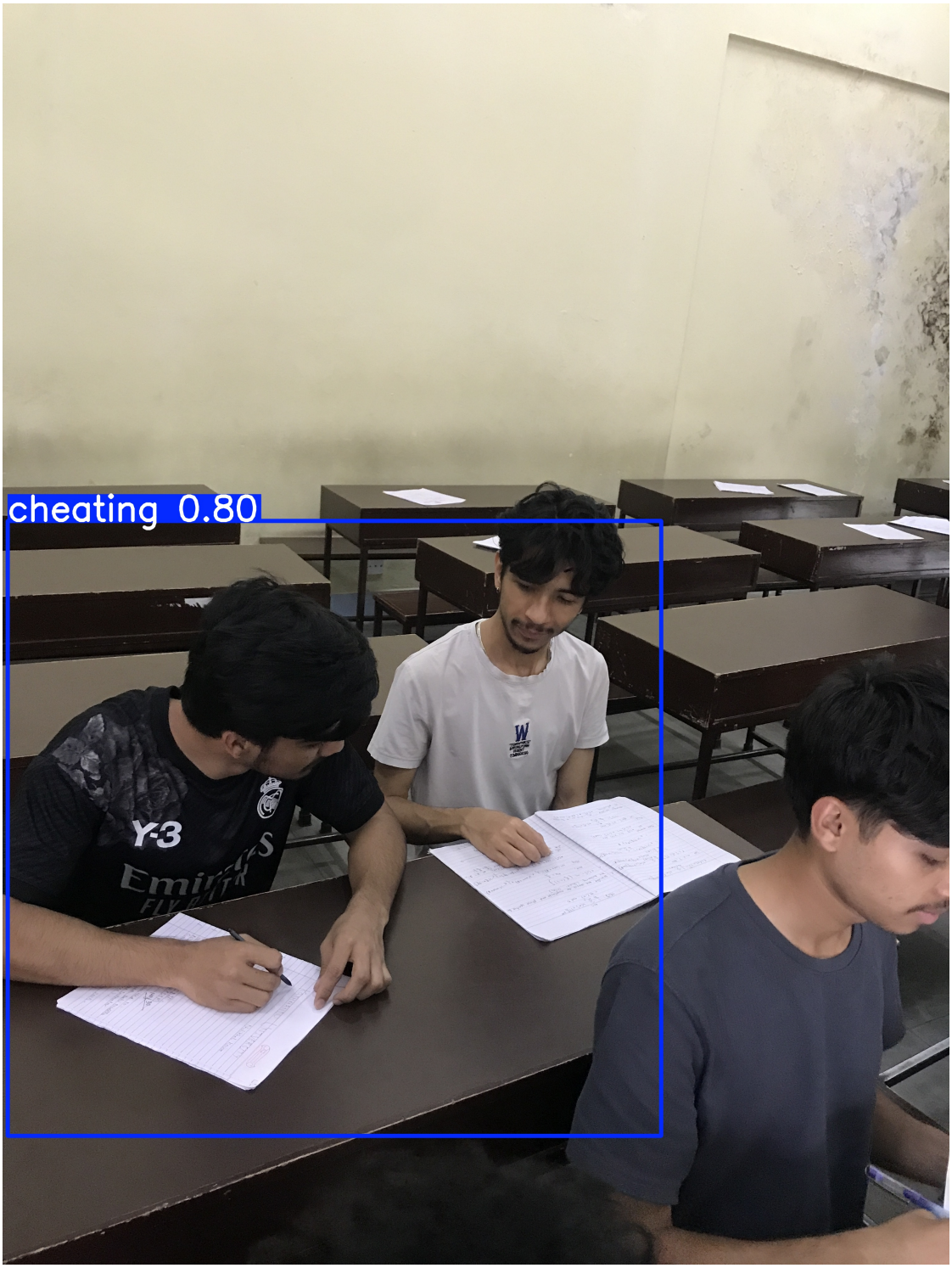

Figure 6-5: Model Output 4

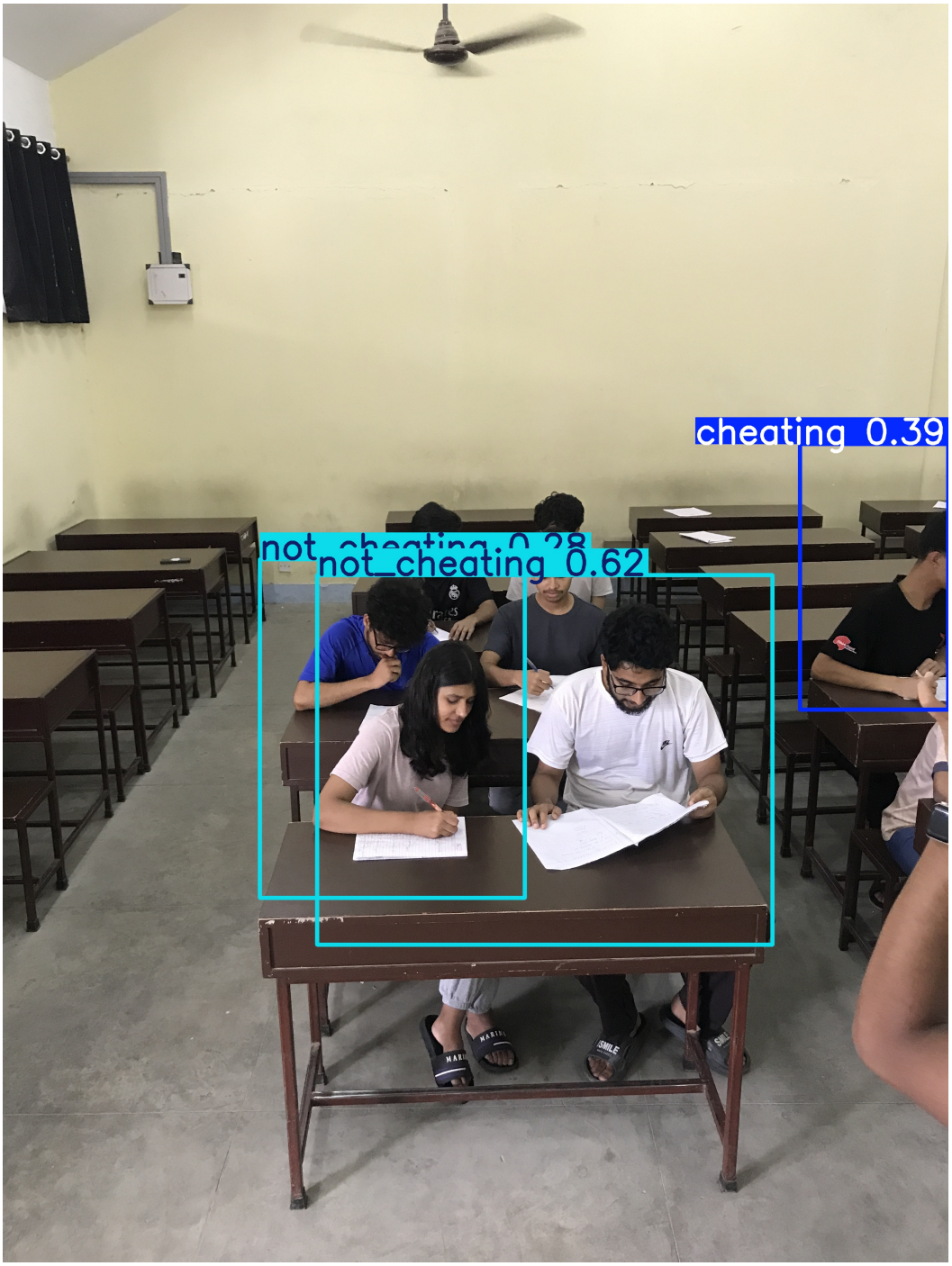

Figure 6-6: Model Output 5

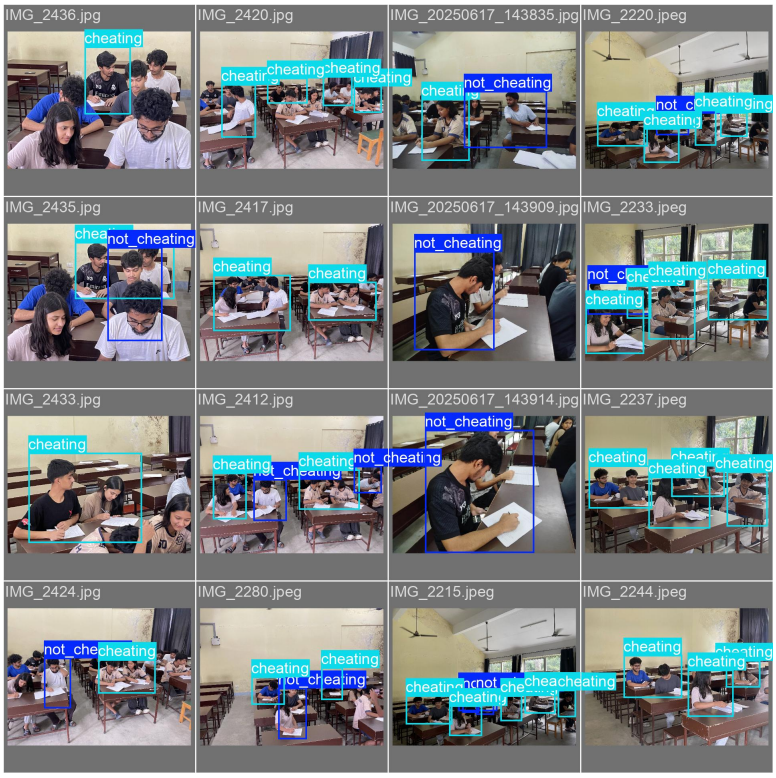

Figure 6-7: Model Results 1

Figure 6-8: Model Results 2

Figure 6-9: Model Results 3

Figure 6-10: Model System 4

Figure 6-11: Model System 5

Figure 6-12: Model System 6

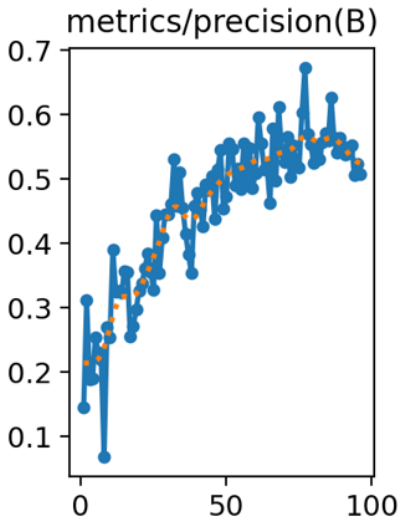

Figure 0-13: Metrics/Precision Curve

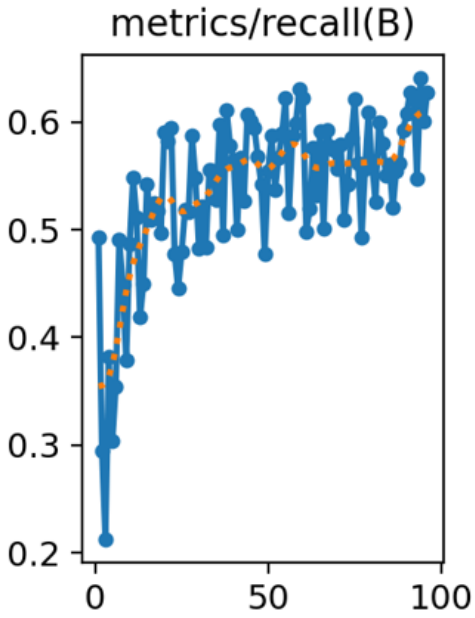

Figure 0-14: Metrics/Recall Curve

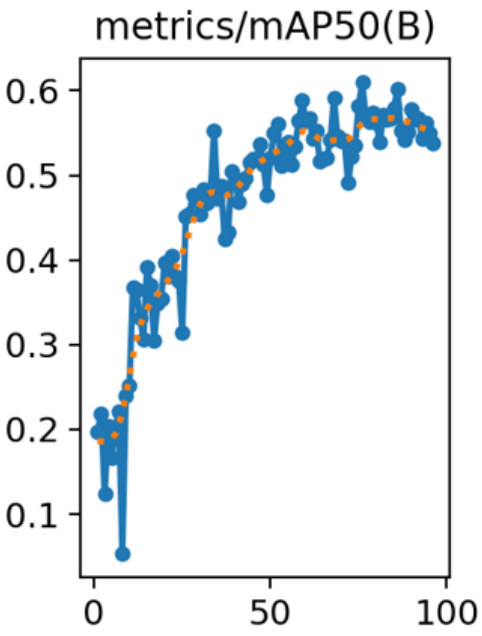

Figure 0-15: Metrics/mAP50 Curve

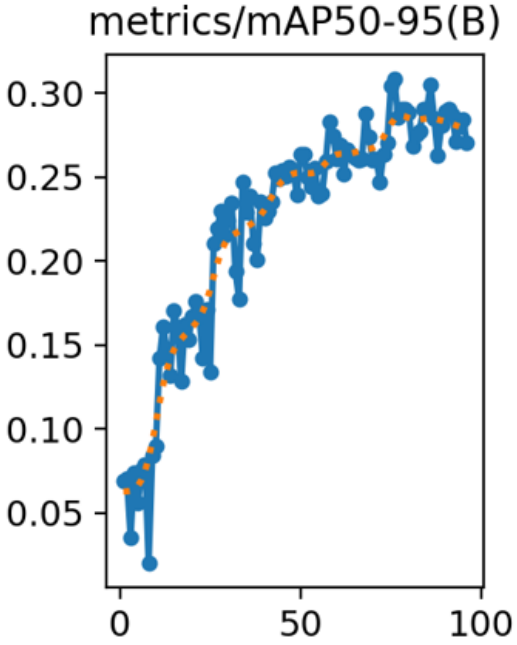

Figure 0-16: Metrics/mAP50-95 Curve

Table 6-1: Metrics Curve Table