جامعة خليفة
Khalifa University

Computer Science Department
COSC 497 | Senior Design Project I
Fall 2024

Midterm report

Project Title:
"Classroom Occupancy Monitoring and Behavior Analysis System"

Submitted to:
Supervisors: Dr. Naoufel Werghi
Dr. Muzammal Nasser
Dr. Maregu Asefa
Examiners: Dr. Panos Liatsis
Dr. Mohammad Alsharid

Submitted by:

Alanood Alharmoodi	**100053854**
Rodha Alhosani	**100058376**
Shaikha Alhammadi	**100058710**
Yasmine Benkhelifa	**100059531**

Table of Contents

Abstract

This report goes over the beginning phases of implementing an AI-driven Classroom Occupancy and Behavior Analysis system designed to enhance safety and optimize the learning experience and resource utilization in educational institutes. The system integrates object detection and behavior analysis using advanced computer vision techniques. The utilization of YOLOv8s was a key component in providing high-speed and accurate detections of students, illegal objects, and defining illegal areas. In the first phase of the project the system is able to detect illegal objects such as beverages in the laboratories and alert unauthorized access in restricted areas. In the second phase, the system will be able to identify disruptive behaviors such as eating as well as accurately detect occupancy levels. Data was collected from CCTV cameras, annotated using MakeSense, and trained using diverse datasets using techniques such as augmentation. A user-friendly dashboard visualizes real-time alerts, logs events, and provides the coordinates of illegal objects. The system went through an intensive validation process, through metrics such as the mean Average Precision (mAP), precision, recall, and more. The innovative solution presented by the system reduces dependency on manual monitoring, minimizes errors, and aids in providing a more productive learning environment.

Keywords: AI-driven system, classroom monitoring, behavior analysis, YOLOv8s, object detection, computer vision, educational safety, illegal area detection, illegal object detection, machine learning.

Chapter 1: Introduction

1.1 Background and Problem Statement

Educational institutes around the world rely on maintaining an environment in classrooms that aid in nurturing future leaders. To provide students and instructors alike with the best possible experience, one must ensure optimal classroom settings. However, issues begin to arise with overcrowding, and disruptive student behaviors such as bringing food or drinks into a lab that can potentially damage costly equipment and pose danger to the wellbeing of others. Relying on traditional methods to monitor classroom occupancy and analyze behavior is time consuming,

subject to error, and inconsistent. These limitations call for a more modern and innovative approach, which makes use of CCTV cameras, automation, and artificial intelligence.

1.2 Goals and Objectives

The primary goal of this project is to develop a sophisticated Classroom Occupancy and Behavior Analysis System that is AI-driven. The team is focused on implementing an innovative solution to an ongoing problem; to enhance classroom management whilst providing a safe and productive learning environment. By implementing an automated approach, the system will allow educational institutes to gain insight on both classroom occupancy and student behaviors, which in turn, will help address and improve educational management practices. The key objectives are as follows:

1. Occupancy Monitoring: Automatically monitor and detect the number of students present in each classroom, to ensure optimal utilization of space and resources.
2. Behavior Analysis: Analyze the behavior of students to recognize and notify signs of distress, disruptive behavior, and partaking in forbidden activities such as eating or drinking in the laboratory.
3. Object Detection: Detect and notify if illegal objects are present anywhere in the classroom, such as bottles or mugs.
4. Illegal Area Detection: Detect and notify if an unauthorized person is standing in the area classified as illegal. This ensures the safety of students and compliance to classroom policies.
5. Safety Enhancement: Enhance the safety of classrooms by detecting unauthorized access and unusual behaviors.
6. User Interface: Provide a friendly user interface for educators that is simple to use that provides instructors with valuable insights such as monitoring classrooms, data visualization, and real-time alerts/notifications.

1.3 Motivation

The motivation behind this project stems from the increasing need for efficient and modern ways to manage classrooms in educational institutes such as universities. The team has recognized the shortcomings of traditional monitoring methods that are inadequate especially in laboratories, where inconsistencies and failure to identify certain behaviors or objects can turn into costly or potentially dangerous situations. Factors such as large classroom sizes, the need for real-time insights, and more require an AI-driven system. Key motivations include:

1. Enhancing Learning Outcomes: Individual and group learning is improved when student engagement is monitored and behavioral issues are identified and improved early on. Traditional monitoring methods often fail to capture behaviors immediately and effectively. Behavior analysis techniques are critical in maintaining classroom order and providing a safe environment [1].
2. Ensuring Safety: Automating the detection of abnormal or illegal behaviors and the presence of an individual in an illegal area enhances safety in classrooms.
3. Reducing Manual Labor: Reducing the dependency on human observers will allow instructors to place their efforts in lecturing. This also reduces the risk of errors and missed detections.

1.4 Methodology

The project follows a structured and systematic methodology approach to ensure that the system is successfully deployed and achieves all of the target goals and objectives. This chapter focuses on giving a brief rundown on the steps the team took, whereas the specifics and details of implementation will be clarified in the upcoming chapters of the report.

1.4.1 System Design and Requirement Analysis

The initial phase included gathering requirements and defining key components to ensure the success of the system. This includes determining which specific behaviors and areas to deem as illegal, and which objects to flag when found present in a classroom. The system architecture was also briefly discussed as well as a high-level overview of the project to define the video processing pipeline.

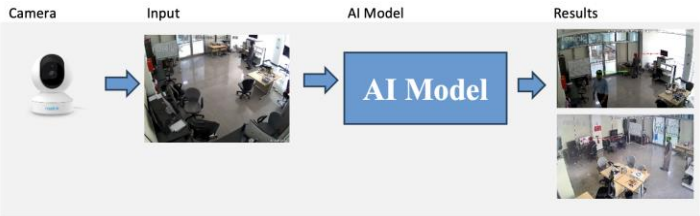


Figure 1 A High-Level Overview Taken from the Project Proposal

1.4.2 Data Collection and Preparation

The system will utilize the existing CCTV cameras that are already installed in classrooms to collect video data. The collected footage includes various scenarios ranging from simple to complex situations to reflect real-life situations. The more simple situations include illegal items placed on desks that are in clear sight, which makes it easy to flag and detect. The more complex scenarios include illegal objects that are further away from the camera and behind or between different objects which make flagging more difficult. The footage collected is then preprocessed using techniques such as frame extraction and noise reduction to prepare the input for model training. This process will be defined and explained in further details in the upcoming chapters.

Figure 2 A Visual of Complex Vs. Simple Situations- The Mugs in Clear View is Easy to Flag Whereas the Black Mug Behind the Black Bag is Difficult to Detect.

1.4.3 Data Annotation

The data annotation phase includes annotating the video data. Frames from the video were used to label and identify illegal objects and areas. The frames were obtained from the training dataset and were manually annotated using the online website MakeSense. Rectangular shapes were utilized to label illegal objects and areas in the frame. Finally, the annotated images were exported as text files in YOLO format, which generated separate label files for each image. The

same process was repeated for validation and testing images. These annotations will serve as ground truth labels for training and evaluating the machine learning model.

1.4.4 Model Selection and Development

The team will make use of object detection algorithms, such as YOLO (You Only Look Once) for real-time detection of illegal objects and areas. While selecting models, factors such as accuracy, efficiency, and real-time performance were taken into consideration. Initially, the team experimented with YOLOv5, where Figure 3 and 4 below shows what objects were able to be detected before any training, fine-tuning, or specifications on which objects to flag. As the project progressed, the team recognized the need for an upgraded model to simplify training. Factors such as the package size of each YOLO model, accuracy, and precision were taken into consideration when deciding on which version to upgrade to. The team then decided on implementing YOLOv8s which provided a good trade-off between model performance and package sizes.

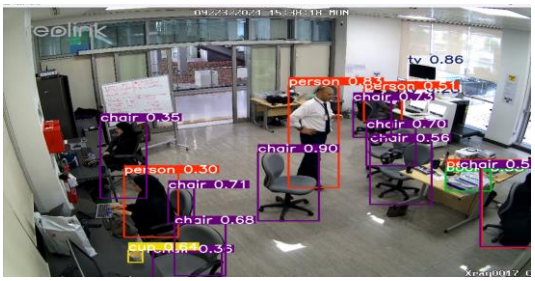


Figure 3 YOLOv5 Detection Prior to any Training or Specifications

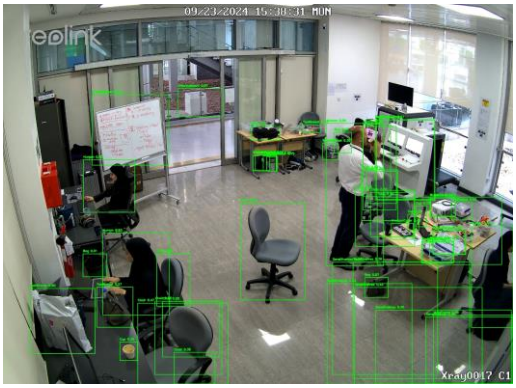


Figure 4 YOLOv8s Detection Capabilities After Minimal Training

1.4.5 Model Training and Evaluation

The annotated dataset is divided into three subsets which include training, testing, and validation. When inputting the training dataset into the model, supervised learning techniques were

used. To assess the model's performance when it comes to detecting illegal objects and areas various evaluation metrics were implemented. Such metrics include F1-score, precision, recall, accuracy, which can be visualized using confusion matrices.

1.4.6 Integrating Computer Vision Techniques

To enhance the detection capabilities of our system, pre-existing computer vision techniques were integrated. This includes making use of algorithms such as object classification and tracking. The purpose of doing this is to ensure our system is functional even in dynamic environments where students are constantly moving around.

1.4.7 System Development and Integration

The next phase involves integrating videos from the CCTV cameras that are installed in classrooms. The pipeline processes the video input in real-time, runs the detection algorithm, and sends the results to a monitoring dashboard.

1.4.8 Real-Time Monitoring

A user interface is included in the system where educators are able to visualize the output of the system. This allows educators to review valuable information that will allow them to make future improvements and enhancements in the classroom. The dashboard will be implemented with key factors in mind such as ease of use, reliability, and simplicity.

1.4.9 Documentation and Deployment

The final phase includes documenting the entire project process, including the steps taken to integrate the model, prepare the data, and develop the model. Finally, the team aims to create a comprehensive user guide and technical documents to assist target users and provide a smooth deployment of the system in educational institutes.

1.5 Overview of the Technical Area

1.5.1 Introduction to Classroom Monitoring Systems

In educational environments it is essential to harbor a safe and structured setting; which is why monitoring student behaviors, classroom occupancy, and objects brought into a classroom is of extreme importance. Manual observations of such behaviors is an inconvenience, and takes valuable time away from the learning process [2]. It becomes especially impractical when certain factors arise, such as large numbers of students, or classrooms with certain restrictions such as illegal areas one should not get close to. With the advancement of technology, there has been a clear shift towards a more innovative solution for such problems, the deployment of automated systems. Incorporating techniques such as computer vision, artificial intelligence, data analysis, and machine learning will allow real-time insights of student behaviors which can be further analyzed by educators [2].

1.5.2 The Current State of Classroom Monitoring Systems

The current state of classroom monitoring systems includes deploying techniques such as manual supervision and basic CCTV cameras. However, there are several limitations found in these methods. In the case of manual supervision, there is a heavy dependency on human observation. Not only is this impractical in big classrooms with a large number of students, it is also inconsistent and prone to error. This is because humans are bound to miss certain disruptive behaviors and cannot detect every object that is brought into the classroom. Additionally, it is labor intensive and is taking away from time that can be spent on things more beneficial to both the students and instructors.

In the case of basic surveillance systems such as standard CCTV cameras, it is beneficial in terms of security, but it does not provide aid to educators and educational institutes. The footage is captured, but there are no real-time insights, object detection, or area detection. Additionally, it cannot analyze classroom occupancy or flag disruptive behaviors. Even with more innovative approaches in the application of CCTV cameras in classrooms, such as the integration of PIR sensors for proactive surveillance [3], the activities still need to be monitored and analyzed manually. For those reasons, there is a growing need for advanced, AI-driven solutions that provide consistent, beneficial analytics beyond capturing videos and manually monitoring students.

1.5.3 Evolution of Automated Behavior Analysis in Classrooms

With the rise of computer vision techniques and machine learning, there has been a significant advancement in the field of automated behavior analysis and object detection. Earlier attempts in the field of simple motion detection involved techniques such as supervised background subtraction. This method makes use of neural networks and supervised learning to improve detection accuracy. However, it relies on objects, so with a growing number of features the complexity significantly increases [4]. Issues will arise from such methods in classrooms where there are large amounts of objects and bodies overlapping.

Currently, rapid developments in deep learning aid in the performance of automated behavior analysis systems. The following techniques play a crucial role in the improvements of this field:

1. Object Detection with Deep Learning: As the team is implementing YOLOv8s, the contributions of YOLO in the object detection and behavior analysis field are of utmost importance. YOLO has the ability to accurately identify numerous objects in one pass. Additionally, the fact that models are pre-trained and can be fine-tuned to specific datasets make YOLO an attractive solution for object detection. The YOLOv8 algorithm is split into three crucial parts: the backbone, neck, and head. This can be visualized with the help of Figure 5 below. The backbone is responsible for feature extraction from the provided input. The neck is responsible for collecting the mappings of features from the backbone. Finally, the head is responsible for the output which includes confusion matrix, confidence scores, and bounding boxes for the detected objects [6].

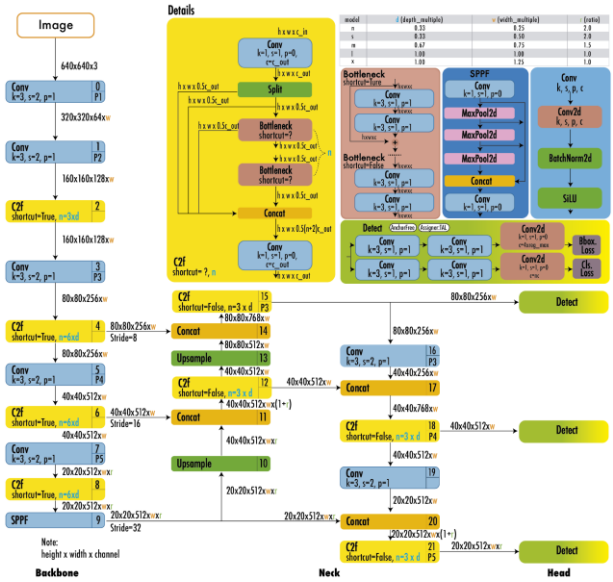


Figure 5 YOLOv8 Architecture [5]

2. Multi-Camera Integration: By integrating footage from various angles into automated behavior systems, deep learning models are able to continuously improve accuracy.

1.5.4 Challenges in Classroom Monitoring

The section above highlights advancements in the field of computer vision and automated systems, however despite these improvements, challenges still exist when it comes to implementing a classroom occupancy and behavior system. These include:

1. Overlapping Bodies: In classrooms, students can overlap with one another or obstruct the visibility of illegal objects. This makes it difficult for a system to capture all illegal objects and behaviors. Additionally, detecting who enters an illegal area becomes more difficult the more crowded a classroom becomes.
2. Varied Lighting Conditions: When the lighting in a classroom is too bright or too dim, it could potentially lead to a negative impact in the performance of a system using computer vision. This requires an intense preprocessing and training phase to prepare a system for such conditions, which could be challenging.
3. Behavior Variability: Detecting illegal behaviors such as eating, drinking, or other disruptive actions in a classroom can vary from student to student, and is context-dependent. The challenge is familiarizing a system with numerous cases of illegal behaviors and ensuring the system can distinguish between when a behavior is illegal and when it is tolerable.

4. Object Variability: Illegal objects such as mugs, water bottles, and food items come in many different shapes and sizes. The system must have previous knowledge of many different food and drink items which can lead to large datasets, which in turn increases difficulty.

1.5.5 Relevant Literature Review

After conducting an intensive literature review on computer vision systems present in educational settings, several key findings were uncovered which include:

1. Attendance Monitoring: New approaches in attendance monitoring that implement face detection algorithms integrated with Learning Management Systems (LMS) are being explored [7]. The objective is to automatically detect and report the attendance of students present in a lecture.
2. Behavior Analysis: Methods are being tested and explored to detect student engagement in classrooms based on factors such as body posture and facial expressions [8]. The objective of such a system is to automatically classify whether students are engaged or not during lectures to aid in the learning process.
3. Occupancy Detection: Implementing YOLO models to count and keep track of occupancy levels inside of a classroom is another way computer vision is implemented in educational settings. The benefits of deploying this in a classroom include improving energy efficiency and adhering to safety protocols [9].

1.5.6 Patent Search

Searching for relevant patents to discover AI-driven technologies targeting behavior analysis and occupancy allow for the identification of unique contributions in this field. It provides a better understanding of existing solutions whilst highlighting what is missing from the market. The research conducted revealed patents related to surveillance systems with behavior analysis, automated attendance systems, and engagement monitoring. However, the methods mentioned do not provide insight into illegal objects and areas, disruptive behaviors, and are not specifically tailored to educational settings. After analyzing existing patents, it was evident that there was a need for a more tailored solution.

1.5.7 Identified Research Gaps

There were several gaps identified despite the advancements in computer vision technologies and automated systems. The first gap noticed is that there is a lack of context-specific solutions, meaning although surveillance systems exist to monitor occupancy they are not catered to classrooms. This could lead to lower detection and accuracy rates since the setting of a classroom is unique and varies greatly from other environments. Additionally, although behavior analysis exists it is mostly aimed at detecting engagement levels by posture and facial expressions. This lacks the detection of disruptive behavior, and activities that are not permissible in a classroom such as eating or drinking. Lastly, the team did not come across a system tailored for detecting whether someone was in an area deemed illegal in a classroom. The lack of this solution is concerning especially in laboratories where entering certain areas could be dangerous or lead to catastrophic outcomes.

1.6 Overview of the Report

The report systematically breaks down the project into key components and tasks, each chapter aligns with the overall goal of creating an AI-driven automated system to monitor classrooms in educational institutes to improve the learning experience. Chapter 1 includes a detailed overview of the development of an advanced system for classroom occupancy and behavior analysis using computer vision and machine learning techniques. A clear outline of the objective of the project along with the goals placed is also found in this chapter. Additionally, the key roles of existing techniques such as YOLO are emphasized. Moreover, the methodology and technical overview including the steps taken to successfully create the automated system. Chapter 2 establishes the system overview and requirement specifications, which serves as a foundation for the following chapters. Chapter 3 dives into an extensive overview of the data construction phase. This chapter includes the details of the data split, data augmentation, and important machine learning concepts such as overfitting. Chapter 4 explores the iterative design process of the project, which includes the steps of development. The training process is also explained further in this chapter. This leads to Chapter 5, where machine learning platforms, technologies used, and data preparation is discussed. Chapter 6 gives a comprehensive report of the validation where results are shown and evaluation metrics are discussed. Finally, the report is concluded with Chapter 7,

where the next steps are discussed, along with a summary of the work done, and challenges the team faced.

Chapter 2: System Overview

2.1 Requirements

The goal of the Classroom Occupancy and Behavior Analysis system is to create an advanced AI-driven system that surpasses traditional observation methods and reduces the need of manual labor. To achieve this, the system follows functional and non-functional requirements that guides the model development process whilst acting as a blueprint.

2.1.1 Functional Requirements

1. Illegal Object Detection: Detect illegal objects in the classroom such as mugs and food items, and clearly state where these objects are detected using x and y coordinates.
2. Illegal Area Detection: Clearly define an illegal area and trigger alerts when an individual enters that premises.
3. Event Logging and Reporting: A system with a friendly user interface should clearly flag and log all detected objects and events based on predefined requirements.
4. Occupancy Monitoring: Accurately detect and count the number of students in a classroom using footage from CCTV cameras. This functional requirement is to be implemented in the next phase of the project.
5. Behavior Analysis: The system must identify disruptive behaviors and flag them, this includes eating or drinking in the classroom. After identifying such behaviors, an alert should be triggered. This functional requirement is to be achieved in the next phase of the project.

2.1.2 Non-Functional Requirements

1. Accuracy: The team aims to achieve high accuracy in student, object, and behavior detection. Part of accuracy is ensuring consistent performances across various classroom setups and environments which is a crucial part in successfully deploying the system. The

team recognizes that this requirement will be solidified further in the next phase of the project.

2. Latency: Having a low latency is important in terms of real-time processing. The team plans to ensure low latency between video input processing and alert generation in the user interface.
3. Scalability: The team aims to ensure the system can handle classrooms of all sizes, with minimal to no degradation in performance even when the number of students increases. This requirement needs further testing in different environments, so it will be greatly improved in the upcoming phases.
4. Robustness: The system should operate effectively with classrooms of different layouts and under various lighting conditions. It should also be able to perform just as effectively even when camera angles change.
5. Usability: An intuitive and friendly user interface should be provided.
6. Efficiency: Video streams should be processed efficiently without using excessive computational resources such as storage.
7. Generalizability: The automated system should be compatible with desktops and laptops of different models.
8. Privacy and Ethics: Ensure that the system collects data and recordings in a way that complies with legal and ethical standards. This will be further explored in the next phase of the project.
9. Cost Effectiveness: In order to ensure that various educational institutes with all types of budgets can gain the benefits of implementing our system, the team aims to balance functionality with affordability.
10. Documentation and Training: When the system will eventually be ready for deployment, user manuals and technical guides must be created to facilitate system adoption.

	Accuracy	Latency	Scalability	Robustness	Usability	Efficiency	Generalizability	Privacy	Cost	Documentation	Score
Accuracy	0	1	1	1	1	1	1	0	1	1	8
Latency	0	0	0	0	0	0	1	0	1	1	3
Scalability	0	1	0	0	0	1	1	0	1	1	5
Robustness	0	1	1	0	1	1	1	0	1	1	7
Usability	0	1	1	0	0	1	1	0	1	1	6
Efficiency	0	1	0	0	0	0	1	0	1	1	4
Generalizability	0	0	0	0	0	0	0	0	1	1	2
Privacy	0	1	1	1	1	1	1	0	1	1	9
Cost	0	0	0	0	0	0	0	0	0	0	0
Documentation	0	0	0	0	0	0	0	0	1	0	1

Figure 6 PCC Table for Non-Functional Requirements

2.2 System Specification

2.2.1 System Architecture

The system follows a modular architecture with the following key components:

1. Input Layer: The existing CCTV cameras installed in classrooms provide video streams to feed into the next layer.
2. Processing Layer: The first step involves videos being preprocessed using OpenCV to enhance clarity and normalize outputs. Next, the trained YOLOv8s model is used to detect students and flag illegal behaviors and objects. Finally, the detected objects and events are logged in order to trigger alerts.
3. Output Layer: Alerts are shown when predefined conditions are met, along with the confidence intervals and coordinates. This is all presented using a friendly user interface.

2.2.2 Hardware Specification

The CCTV cameras should have a minimum resolution of 1080p to ensure high quality video outputs that are clear, in order for our model to accurately process the input. Additionally, sufficient storage space is required in order to flag, download, and look back at previously processed videos.

2.2.3 Software Specification

Machine learning models are being made use for object and behavior detection. YOLOv8s allows students and objects flagged as illegal to be detected after completing an intensive training process. The next phase of the project will require flagging illegal behaviors. Various frameworks

and libraries such as OpenCV, Pytorch, and more are used to develop and train the model. The key technologies used are mentioned below:

1. **OpenCV**

   OpenCV was essential in the project, enabling real-time video stream processing and frame analysis. It seamlessly integrates with YOLO to handle object detection and tracking, making it a cornerstone of the system's functionality.

2. **Ultralytics**

   The Ultralytics library simplified the implementation of YOLOv8. Providing pre-trained models and an adaptable framework for fine-tuning accelerated development, allowing the team to customize models effectively for the specific use case.

3. **Tkinter**

   Tkinter was used to create a user-friendly graphical interface that allows users to interact with the system intuitively—whether it's for live monitoring, receiving alerts, or managing system settings. Figure 7 showcases an example of the Tkinter interface, illustrating its simplicity and effectiveness.

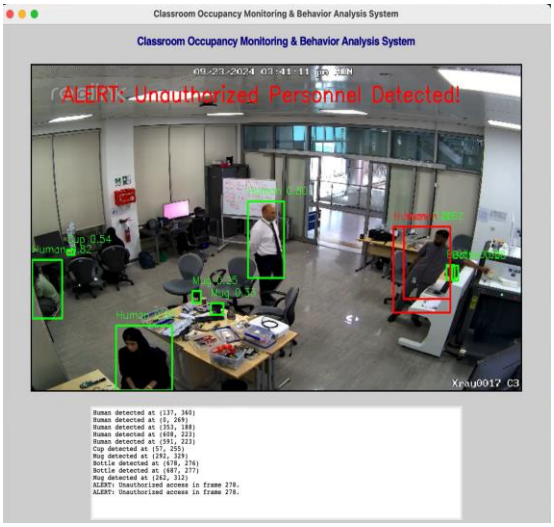


Figure 7 Showcasing the Simplicity of Tkinter

4. **PyTorch (Python 3.10.15)**

   PyTorch served as the backbone of the model training process. The advanced features for managing large-scale datasets and resource-intensive tasks made it the ideal choice for our needs. By leveraging GPU acceleration, training times were significantly reduced through parallel computations, which was essential given the size of the data.

   Furthermore, PyTorch's dynamic computation graph provided the flexibility needed to experiment with various model architectures and refine training strategies efficiently. Its compatibility with Python 3.10.15 ensured smooth integration with other tools and libraries [10].

By integrating these advanced tools and methodologies, the system will be designed to deliver insights that enhance safety and improve the learning experience across educational institutes for instructors and students alike.

Chapter 3: Dataset Construction

The dataset construction phase is crucial for the system's success because the model's performance depends on the quality and variety of the data. This chapter explains how the dataset was created and prepared to achieve the project's goals. Each step was done carefully to ensure that the dataset can support effective object detection, allowing the system to work well in real-world situations.

3.1 Data Configuration

The team began by defining the dataset configuration in a data.yaml file. This file specified the number of classes (nc) and their corresponding labels (names). This configuration served as the foundation for training the YOLOv8 model. Below is the configuration the team used:

```yaml
nc: 7
names:
  0: Cup
  1: MainFrame
  2: Bottle
  3: Human
  4: Snack
  5: Mug
  6: Biscuit
```

Figure 8 Data.yaml classes

This setup was designed to capture a diverse set of objects, with "MainFrame" representing the restricted area. These classes reflected the specific objects and behaviors the team aimed to monitor and flag in the classroom environment.

3.2 Data Acquisition

3.2.1 Data Collection:

The team began by collecting data from the lab environment using both installed cameras. This setup ensured that the dataset captured real-world conditions specific to the lab, including varying angles, lighting conditions, and object arrangements. To further enhance the dataset, additional labeled data was sourced from Roboflow. This comprehensive approach allowed the team to build a robust dataset reflective of both the specific lab environment and more general real-world scenarios.

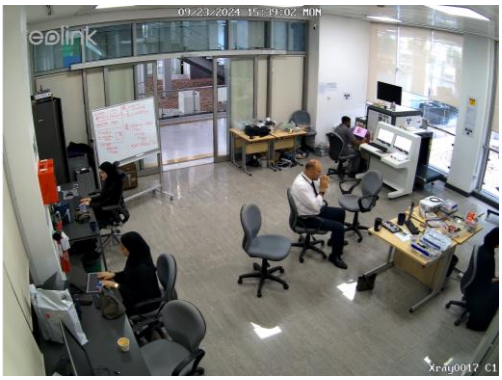


Figure 9 Data Collection from Multiple Cameras in the Lab

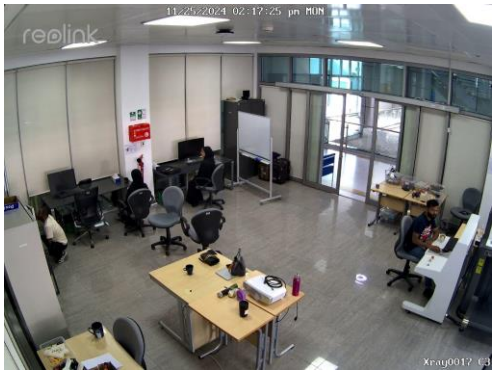


Figure 10 Data Collection from Multiple Cameras in the Lab

3.2.2 Incorporating Roboflow Datasets

To enhance the training dataset further, the team integrated additional data sourced from Roboflow. These datasets contained labeled images of objects such as cups, mugs, and bottles in various shapes, colors, and environments. Roboflow provided high-quality, pre-labeled datasets in YOLOv8 format, saving significant time in the annotation process. The variety in Roboflow datasets allowed the model to generalize better to unfamiliar conditions. The impact of adding dataset from Roboflow, is by adding Roboflow data to the augmented training dataset, the team ensured the model was less likely to underfit, particularly when encountering objects in new settings or with unfamiliar variations.

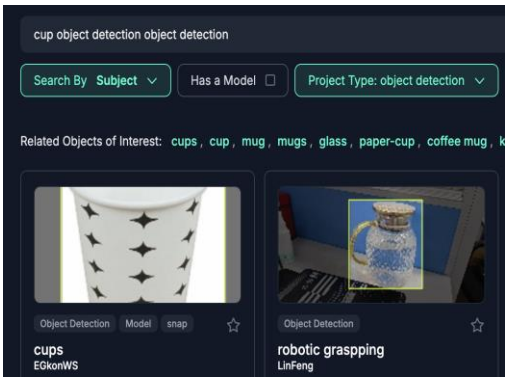


Figure 11 Datasets collected from Roboflow

3.2.3 Object-Specific Data Inclusion and Augmentation

Case 1: Cup

Reason for Inclusion:

Cups are a frequently occurring object in the lab environment, varying in design, size, and material. Accurate detection of cups across diverse scenarios is vital for the system's effectiveness.
Data Augmentation:

To improve the model's ability to generalize across different situations, the team incorporated cups in various:

- **Colors:** Red, blue, white, and transparent to reflect variations in appearance.
- **Shapes and Sizes:** Examples included large mugs, small coffee cups, and disposable cups.
- **Orientations:** Rotations (e.g., 45°, 90°) simulated real-world viewing angles.
- **Lighting Conditions:** Frames with both dim and bright lighting conditions were added.
- **Background Variability:** Frames included cluttered tables, clean surfaces, and cups partially occluded by other objects to mimic challenging scenarios.

Case 2: MainFrame (Restricted Area)

Reason for Inclusion:
The 'MainFrame' represents a restricted area requiring constant monitoring to ensure no unauthorized access occurs. Accurate detection of humans entering this zone is crucial.
Data Augmentation:

- **Perspective Changes:** The MainFrame zone was labeled from various camera angles to provide diverse viewpoints.
- **Lighting Variability:** Frames with both bright and dim lighting conditions were added to simulate different times of day.
- **Edge Cases:** Examples where the MainFrame boundary was partially obscured by objects were included to test detection robustness.

Case 3: Bottle

Reason for Inclusion:

Bottles are commonly found in the lab and can be confused with similar items like glasses or jugs.

Data Augmentation:

- **Class Consistency:** All water bottle labels were remapped to 'Bottle' for uniformity.
- **Shapes and Sizes:** Included cylindrical bottles, flat bottles, and sport-style bottles.
- **Rotations:** Examples of bottles lying on their side, upright, or tilted.
- **Color Variations:** Transparent, green, and blue bottles were added

Case 4: Human

Reason for Inclusion:

Humans are the primary entities for monitoring. YOLOv8's pre-trained models already exhibit high accuracy in human detection.

Enhancements:

- Video frames from the lab environment were integrated to ensure reliable detection of humans in the specific context.
- Varied postures such as standing, sitting, and bending were included to reflect real-world scenarios.

Case 5: Mug

Reason for Inclusion:

Mugs, while similar to cups, differ in shape, size, and function. Including mugs as a separate class ensures precise detection.

Data Augmentation:

- **Diverse Designs:** Examples included plain, patterned, and branded mugs.
- **Handle Variations:** Mugs with handles oriented to the left, right, or hidden due to occlusion were added.
- **Perspective Changes:** Frames captured mugs from top-down, side-on, and angled views to simulate real-world appearances.

By addressing each object class with tailored data augmentation and diversity, the team ensured a balanced and comprehensive dataset. This approach enhances the detection model's ability to generalize and perform reliably in dynamic, real-world environments. The inclusion of the 'MainFrame' restricted area demonstrates the system's dual capabilities in object detection and spatial monitoring, making it a robust tool for lab environments.

Figure 12 Visual Representation of Data Augmentation

3.2.4 Feature Engineering Techniques

In our project, feature engineering was crucial in optimizing the performance of the YOLOv8 model for real-time object detection and tracking in video feeds. Several feature engineering techniques were used to extract meaningful information from the video frames and improve detection accuracy:

1. Bounding Box Features: The bounding box coordinates, which define the position and dimensions of detected objects, were central to our feature extraction. These coordinates were essential for localizing objects in each frame. Additionally, attributes like the area and aspect ratio of the bounding boxes were derived to help differentiate objects with varying sizes and shapes.
2. Confidence Scores: YOLOv8 outputs a confidence score for each detected object, reflecting the model's certainty about the presence of the object within a bounding box.

These confidence scores were utilized as a feature to filter out detections that were less reliable, prioritizing high-confidence predictions for further analysis.

3. Object Class Labels: The model's predicted object class labels, such as person, car, or animal, were used to categorize detected objects. These class labels were important for distinguishing between different object types, enabling the system to apply relevant processing techniques depending on the object detected.

3.2.5 Overfit and underfit

Overfitting and underfitting are key challenges in machine learning that can limit a model's ability to generalize effectively to unseen data. Overfitting occurs when a model becomes too tailored to the training data, memorizing noise and irrelevant details instead of identifying meaningful patterns. This results in excellent performance on the training data but poor results when applied to new, unseen data. On the other hand, underfitting happens when a model is too simplistic to recognize the underlying patterns in the data, leading to poor performance on both the training set and new data.

Striking a balance between overfitting and underfitting is crucial for building a model that generalizes well. This involves finding the right level of complexity in the model to ensure it learns relevant features while avoiding unnecessary details. Generalization—the ability to apply what the model has learned from the training data to unseen data—depends heavily on the quality and diversity of the dataset, as well as the model's architecture and training process [11].

To tackle these challenges, the team utilized a carefully curated and diverse dataset, which combined data collected in a lab environment with high-quality, pre-labeled datasets from Roboflow. The addition of lab video frames allowed the model to capture real-world scenarios with unique lighting conditions, camera angles, and specific classroom objects. The Roboflow dataset contributed variety by including items such as cups, mugs, and bottles of different shapes, sizes, and colors, captured in various environments.

This dual-source approach created a balanced dataset that reduced the risks of overfitting and underfitting. By combining structured data (from Roboflow) and unstructured data (from lab video frames), the team ensured that the model could generalize effectively to new, unseen data.

Moreover, data augmentation techniques were employed to further diversify the dataset, preventing the model from memorizing training examples and enhancing its adaptability to real-world situations. This methodology established a solid foundation for achieving a model that performs reliably across diverse conditions and complex environments.

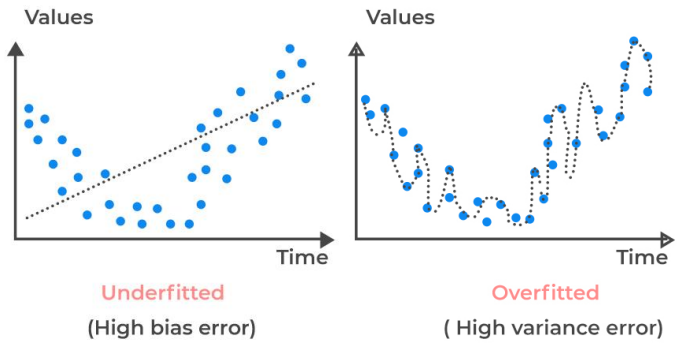


Figure 13 Visual Representation of Underfitting and Overfitting

3.3 Data Augmentation

To increase the diversity and volume of the training dataset, the team applied data augmentation to the extracted frames. Augmentation is a critical step that ensures the model is exposed to a variety of scenarios, improving its generalization capability and robustness against unseen conditions.

Frame Extraction

The team developed a Python script to systematically extract 18 frames from each video at regular intervals. This approach was carefully designed to capture diverse moments within the video without introducing redundancy. By doing so, the dataset retained the uniqueness of each instance while covering various angles, lighting conditions, and object positions within the videos. Each extracted frame underwent one randomly selected augmentation from a set of transformations, which included rotation, flipping, brightness adjustment, scaling and cropping.

Why Augmentation Was Applied

1. **Prevent Overfitting**: Augmentations exposed the model to diverse scenarios, reducing the risk of memorizing training data and instead encouraging generalization.

2. **Improve Real-World Adaptability**: Augmentation ensured that the model could detect objects across varying conditions such as lighting changes, different object orientations, and partial occlusions.
3. **Expand Dataset Volume**: The transformations increased the dataset size without requiring additional data collection efforts, creating a cost-effective and efficient way to enhance model performance.
4. **Enhance Detection Accuracy**: By training on augmented images, the model's capacity to recognize objects in unpredictable settings was significantly improved.

Impact of Augmentation

The application of augmentation techniques substantially increased the diversity of training examples. This played a crucial role in making the model more resilient to real-world complexities, such as detecting cups, bottles, and restricted areas in dynamic environments with variations in perspective, lighting, and occlusions. Ultimately, this step helped the system achieve higher detection accuracy and adaptability, even in challenging and unstructured settings [12].

3.4 Data Labeling

The data labeling process was conducted meticulously to ensure the highest quality annotations for training the object detection model. The steps were as follows:

1. **Uploading Frames:** Extracted frames from the dataset were uploaded to the MakeSense.AI interface, an intuitive tool designed for efficient data annotation.
2. **Manual Labeling:** Team members manually labeled each frame by drawing bounding boxes around objects or areas of interest. Each bounding box was assigned the corresponding class label, such as "Cup" or "MainFrame", ensuring accurate categorization of all detected entities.
3. **Exporting Annotations:** The completed annotations were saved in the YOLO format, a lightweight structure that integrates seamlessly into the training pipeline for object detection models.

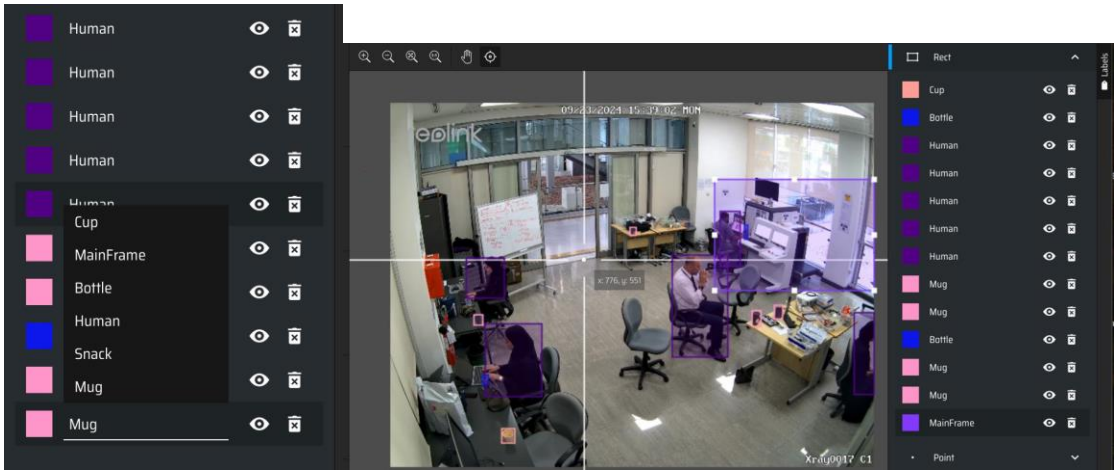


Figure 14 Manual Annotation Using MakeSense

This manual labeling process ensured the annotations were not only accurate but also consistent across the dataset. These high-quality annotations were vital for optimizing the performance of the YOLOv8 model, allowing it to learn effectively from the labeled data and achieve robust detection capabilities in real-world scenarios.

3.5 Training, Validation, and Testing Split

To ensure the model's generalization and reliability, the dataset was divided into Training, Validation, and Testing sets with the following distribution:

- **Training Set (70%):**
  This set was used to train the model by providing labeled images for parameter optimization. The large proportion of training data ensured that the model could learn effectively from diverse examples.

- **Validation Set (20%):**
  During training, this set was used to monitor the model's performance and fine-tune hyperparameters. Validation helped prevent overfitting by providing an independent dataset to assess the model's adaptability to unseen data.

- **Testing Set (10%):**
  Reserved exclusively for final evaluation, the testing set allowed the team to measure the

model's performance on entirely unseen data. This phase provided an objective assessment of the model's accuracy, precision, and recall in real-world scenarios.

This distribution ensured that the model had sufficient data for learning while maintaining dedicated subsets for validation and testing, resulting in a robust and generalizable detection system.

Chapter 4: Steps of Development

4.1 Steps of Development

Machine learning is the key technology for this project, serving as the foundation for real-time object detection and behavior analysis. The development process follows a structured pipeline that begins with problem definition, followed by data collection and annotation. The core of the system is built on the YOLOv8 framework, known for its high speed and accuracy in detection. The essential steps include model training, validation, and testing, which ensure the system's reliability and ability to adapt to dynamic environments. This approach allows for effective object detection and alert mechanisms, even in complex scenarios.

4.2 Problem Definition

The system is designed to address two primary challenges:

- **Real-time Object Detection and Classification:** Accurately identifying and classifying objects in a dynamic lab environment to ensure reliable monitoring of key items such as cups, bottles, and mugs.
- **Detection of Unauthorized Proximity to Restricted Areas:** Monitoring and identifying breaches of designated restricted zones (e.g., the "MainFrame") to ensure compliance with access protocols and enhance safety.

4.3 Model Selection

YOLOv8 is a state-of-the-art deep learning framework for object detection, chosen for its real-time performance and high accuracy. Its capability to handle multiple objects in a single frame makes it ideal for dynamic environments like laboratories. During the development phase, the team also evaluated YOLOv5 as a potential model. Although YOLOv5 showed good performance, its detection accuracy and speed were not as robust as those of YOLOv8 when applied to our dataset, especially for smaller object categories such as "Cup" and "Mug," or in cases of overlapping objects. YOLOv8's anchor-free architecture and enhanced dynamic detection head contributed to better generalization and improved performance in these situations. Therefore, YOLOv8 was selected as the final model for this project.

Figure 15 compares the performance of different versions of YOLO (You Only Look Once) in terms of latency and accuracy on the COCO dataset. The graph plots the COCO mAP50–95 (mean Average Precision at IoU thresholds from 0.5 to 0.95) against the latency, measured in milliseconds per image, for various YOLO models. The blue line represents YOLOv8, which consistently outperforms the earlier versions (YOLOv7, YOLOv6-2.0, and YOLOv5-7.0) in terms of accuracy while maintaining lower latency [13].

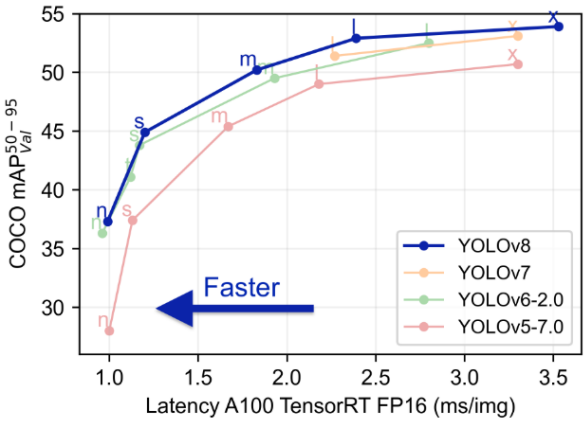


Figure 15 Performance of Various YOLO Models

4.4 Training Process

The training process was designed with careful consideration of the dataset's structure, model requirements, and the computational resources available. The goal was to develop a robust object detection system capable of accurate predictions across diverse scenarios. This section outlines the key elements of the training process, including the rationale behind the chosen configurations.

4.4.1 Training Data Split

A significant portion of the data, 70%, was allocated for training to ensure the model had sufficient exposure to varied examples. This decision was made to enable the model to learn a wide range of patterns, object variations, and scenarios effectively.

4.4.2 Batch Size and Epochs

- **Batch Size:** The batch size was set at 16 to optimize GPU utilization without risking memory overflow. This size provided a balance between computational efficiency and stability during convergence.
- **Epochs:** The model was trained for 60 epochs, a number determined after initial experimentation. This duration allowed the model adequate time to learn from the data without overfitting. Early stopping mechanisms based on validation metrics were employed to halt training if the model began to exhibit signs of overfitting. The choice of 60 epochs ensured sufficient iterations for effective convergence, especially for larger objects like "Human" and "MainFrame," while highlighting areas for improvement in detecting smaller objects such as "Cup" and "Mug."

4.4.3 Training Workflow

1. **Model Initialization:**

   The team leveraged a combination of pre-trained COCO weights, a custom lab-collected dataset, and data from Roboflow to enhance robustness and adaptability. The lab-collected dataset comprised video frames annotated to represent real-world classroom scenarios, incorporating unique camera angles, lighting conditions, and the "MainFrame" restricted

area. This ensured the model was tailored to its specific use case. The integration of Roboflow datasets, which included objects like cups, mugs, and bottles in various shapes, sizes, and environments, added diversity and improved the model's ability to generalize. This multi-source approach minimized the risks of overfitting and underfitting, creating a strong foundation for real-world application.

2. **Loss Function Optimization:** The model's training process optimized a composite loss function comprising [14]:
    - **Bounding Box Loss:** Improved object localization precision.
    - **Classification Loss:** Reduced errors in object classification.
    - **Objectness Loss:** Enhanced confidence in detecting actual objects while suppressing false positives.
3. **Learning Rate Scheduling:**A cyclical learning rate scheduler dynamically adjusted the learning rate during training. This approach balanced the exploration of new parameter spaces with the refinement of existing solutions, facilitating faster convergence.
4. **Augmentation and Preprocessing:**Data augmentation techniques such as random rotations, flips, brightness adjustments, and cropping were applied to enhance generalization. These augmentations exposed the model to a variety of real-world variations, improving its adaptability [15].
5. **Evaluation During Training:**Validation metrics, including mean Average Precision (mAP), precision, and recall, were monitored after each epoch. These metrics provided feedback to guide adjustments in hyperparameters such as learning rate and augmentation probabilities.

Results and Observations

- **Performance on Larger Objects:** The model demonstrated high accuracy for larger objects such as "Human" and "MainFrame," attributed to their prominence in the dataset and the robustness of the YOLOv8 architecture.
- **Challenges with Smaller Objects:** Detection rates for smaller items like "Cup" and "Mug" required additional fine-tuning and augmentation to enhance accuracy.

The training process successfully utilized a well-structured dataset, optimized hyperparameters, and robust augmentation techniques to achieve strong detection capabilities. By allocating 70% of the data for training and carefully monitoring validation metrics over 60 epochs, the team developed a balanced and reliable model capable of generalizing to real-world scenarios.

4.5 Validation Process

The validation process was integral to evaluating the model's performance and guiding improvements during training. The validation set, comprising 20% of the dataset, served as a benchmark to measure how well the model generalizes to unseen data. Key metrics such as Mean Average Precision (mAP), Precision, and Recall were assessed using YOLOv8's built-in evaluation tools. These metrics provided a comprehensive view of the model's accuracy in detecting and classifying objects, as well as its ability to localize objects effectively within frames.

4.5.1 Iterative Refinement

Feedback from the validation process enabled iterative enhancements to improve the model's performance. Specific refinements included:

- **Underperforming Classes:** Categories like "Mug" and "Cup," which initially exhibited lower detection accuracy, were targeted with focused data augmentation. Additional examples were generated by applying transformations such as rotation, scaling, and brightness adjustments to enhance the model's ability to recognize these objects in diverse conditions.
- **Hyperparameter Tuning:** The team fine-tuned critical parameters, including:
    - **Learning Rate:** Adjusted dynamically to balance between rapid convergence and stable learning, following cyclical patterns to optimize parameter updates.
    - **Batch Size:** Experimented with variations to optimize GPU utilization while avoiding memory overflow.

4.5.2 Impact of Validation and Refinement

The iterative validation process ensured that the model remained adaptable and resilient, effectively reducing overfitting and underfitting. Observations during validation guided adjustments that:

- Improved mAP scores across all object classes.
- Enhanced the precision and recall metrics, particularly for smaller and less frequent classes like "Mug" and "Cup."
- Ensured the model's performance was stable across diverse scenarios, including varying lighting conditions and occlusions.

By leveraging validation feedback and systematically refining the model, the team developed a robust object detection system capable of reliable performance in real-world settings.

4.6 Testing Process

The testing phase was conducted using 10% of the dataset, specifically reserved to evaluate the model's ability to generalize and perform in real-world scenarios. This portion of the dataset comprised unseen videos, ensuring an unbiased assessment of the model's robustness and detection accuracy.

Key Objectives of Testing

1. **Robust Object Detection:**
    - The model's ability to detect multiple objects such as "Human," "Cup," and "MainFrame" in dynamic and real-time settings was evaluated.
    - Detection accuracy for both prominent ("Human," "MainFrame") and smaller objects ("Cup," "Mug") was monitored to ensure consistent performance across categories.
2. **Triggering Alerts:**
    - The IoU-based mechanism was tested to detect unauthorized access to restricted areas ("MainFrame").
    - Alerts were verified to trigger whenever a "Human" entered the restricted zone, with corresponding frames automatically saved for further analysis.

3. **Performance under Variability:**
    - Videos with diverse lighting conditions, such as bright and dim environments, were included to evaluate robustness under real-world conditions.
    - Frames with occluded or partially visible objects were tested to confirm the model's adaptability to challenging scenarios.

Metrics Captured During Testing

1. **Precision and Recall:**
    - Precision was assessed to ensure that identified objects were relevant and accurate, minimizing false positives.
    - Recall was evaluated to ensure the model detected most of the actual objects, minimizing false negatives.
2. **False Positive Rate:**
    - The frequency of unnecessary alerts was monitored to ensure the system did not become overly sensitive, maintaining practical usability.
3. **Consistency Across Frames:**
    - Stability of detections in consecutive video frames was ensured, reducing fluctuations and false triggers in alerts.

Observations and Insights

- **Performance on Larger Objects:**
    - The model demonstrated high accuracy in detecting prominent objects like "Human" and "MainFrame," even under varied lighting and partial occlusions.
- **Challenges with Smaller Objects:**
    - Smaller objects like "Cup" and "Mug" occasionally faced missed detections, requiring further refinement in data augmentation and hyperparameter tuning.
- **Alerting System:**
    - The IoU-based mechanism effectively detected and flagged unauthorized access to restricted areas, validating its utility for spatial monitoring.
- **Real-Time Capability:**

- The system maintained an average inference speed of 30ms per frame, enabling smooth and efficient real-time performance.

The testing phase validated the model's ability to generalize effectively to unseen data and perform robust object detection in complex real-world conditions. By capturing detailed metrics such as precision, recall, and false positive rates, the team ensured that the system could reliably detect objects, issue accurate alerts, and maintain performance consistency in real-time applications. The testing insights further informed refinements in the model, including targeted adjustments for smaller object categories and fine-tuning the alerting mechanism.

Chapter 5 Validation

5.1 Evaluation Metrics

Evaluation metrics provide a comprehensive overview of a model's performance across various object detection tasks. These metrics assess the system's ability to detect and classify objects accurately while minimizing false positives and false negatives. Key metrics such as precision, recall, and mean Average Precision (mAP) offer a quantitative evaluation of the model's effectiveness. Confusion matrices further break down performance by class, identifying specific areas for improvement. Additionally, inference speed metrics validate the system's ability to operate efficiently in real-time, ensuring it meets the project's performance objectives [16].

1. **Precision and Recall:** Precision and recall are fundamental metrics used to measure the model's detection accuracy. Precision evaluates the proportion of true positive detections among all positive predictions, while recall assesses the proportion of true positive detections among all actual positives. These metrics are particularly critical in reducing false alarms and ensuring all relevant objects, such as humans in restricted zones, are detected [17].

Equation 1 Precision

Equation 2 Recall

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

2. **Mean Average Precision (mAP):** Evaluates the model's performance across all object categories by averaging the precision-recall curve for each category. mAP is an essential metric for multi-class object detection tasks, providing a holistic view of the system's effectiveness [18].

Equation 3 Mean Average Precision (mAP)

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i$$

Where N is the number of categories and AP_i is the average precision for category i.

The difference between the mAP@0.5 and mAP@0.5:0.95 is the different Intersection over Union thresholds, where one is a 50% overlap and the other averages across 0.50 to 0.95% overlap. The distinction helps detect the robustness in detecting objects over different IoU levels.

3. **Confusion Matrix**: Confusion matrices offer a detailed breakdown of classification performance for each object class, highlighting true positives, false positives, true negatives, and false negatives. These matrices help the team identify underperforming categories (e.g., smaller objects like "Cup" and "Mug") and guide targeted refinements, such as additional data augmentation or hyperparameter tuning [19].

The table below outlines the key metrics used in the confusion matrix to evaluate how well the model performs in classification tasks. These metrics compare the predicted outcomes

with the actual labels, helping to identify areas where the model succeeds or needs improvement:

Table 1 Description of Confusion Matrix Metrices

Metric	Description
True Positive TP	An instance for which both predicted and actual values are positive.
False Positive FP	An instance for which predicted value is positive but actual value is negative.
False Negative FN	An instance for which predicted value is negative but actual value is positive.
True Negative	An instance for which both predicted and actual values are negative.

4. Object Detection Visualization: The image demonstrates the system's real-time object detection capability, where multiple objects such as "Human," "Cup," "Mug," and the "MainFrame" restricted area are accurately identified and labeled with confidence scores. This visualization helps in monitoring activities and ensuring compliance with restrictions, as objects of interest are highlighted for easy interpretation and quick decision-making.

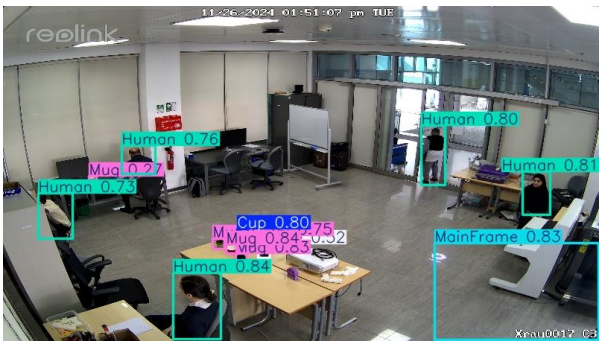


Figure 16 Object Detection Visualization

5.1.1 Results

This section provides an in-depth analysis of the model's performance across various stages, including training evaluation, testing evaluation, and failure analysis. The training evaluation focuses on metrics such as precision, recall, and mean Average Precision (mAP),

assessing how well the model learned from the dataset. The testing evaluation examines the model's ability to generalize to unseen scenarios, highlighting its robustness in challenging conditions. Finally, the failure analysis identifies areas where the model struggled, offering insights into potential improvements for future iterations.

5.1.2 Training evaluation

The evaluation of the model's performance during training and testing is crucial to understanding its ability to generalize and accurately classify objects. Several metrics and visualization tools, such as confusion matrices, were utilized to provide a detailed analysis of the model's strengths and weaknesses.

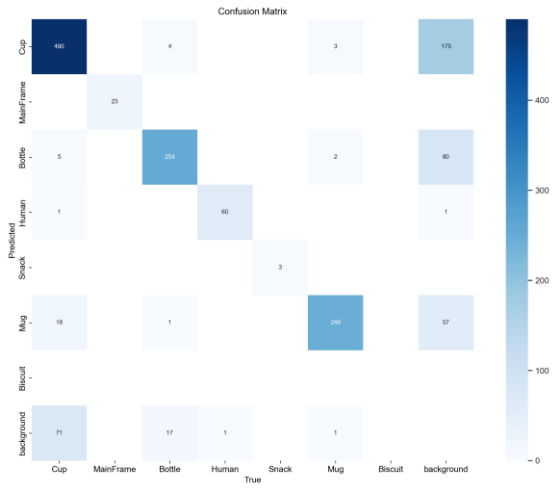


Figure 17 Confusion Matrix After Training YOLOv8s Model

Confusion Matrix Analysis

The confusion matrix, as shown in figure 17 above, offers a class-wise breakdown of predictions, highlighting True Positives (correct predictions) on the diagonal and misclassifications in the off-diagonal values. Below are the observations based on the raw and normalized confusion matrices:

1. **Cup Class**:
    - Correct Predictions: 490 True Positives.

- Misclassifications: 175 instances were labeled as "Background," indicating the model occasionally struggles to distinguish cups in cluttered or ambiguous scenes.

2. **MainFrame Class**:
   - Excellent performance with **100% accuracy**. All 23 instances were correctly identified, demonstrating the model's ability to detect restricted areas reliably.
3. **Bottle Class**:
   - Correct Predictions: 254 instances.
   - Misclassifications: 80 instances were categorized as "Background," suggesting the need for further augmentation to distinguish bottles from background objects in complex settings.
4. **Human and Snack Classes**:
   - Strong performance with minimal misclassifications. This indicates that the model is well-trained to detect humans and snacks even in challenging environments.
5. **Mug Class**:
   - Correct Predictions: 246 instances.
   - Misclassifications: 57 labeled as "Background," highlighting the need for additional training or augmentation to handle variations in mug appearances or settings effectively.

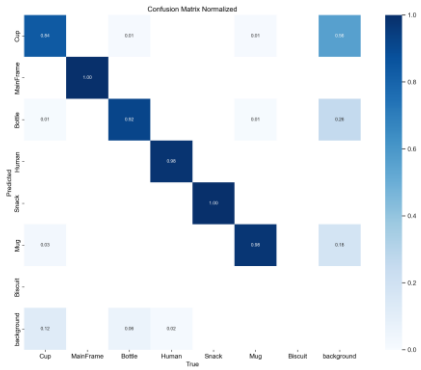


Figure 18 Normalized Confusion Matrix

Normalized Confusion Matrix

The normalized confusion matrix in figure 18, which displays the proportion of predictions rather than raw counts, provides a more intuitive understanding of the model's reliability for each class. For instance:

- **MainFrame Class**: A perfect score of **1.00**, confirming the model's consistency for restricted area detection.
- **Cup Class**: Despite achieving a precision of 0.84, the model showed some confusion with the background (0.56 misclassification rate), necessitating further refinement.
- **Mug Class**: Precision of 0.98, although occasional overlap with the background exists.

Observations and Recommendations

- **Overall Performance**: The model performed exceptionally well for larger and more distinct objects, such as "Human" and "MainFrame."
- **Smaller Objects**: Items like "Cup" and "Mug" require additional augmentation strategies, such as variations in orientation, occlusion, and lighting conditions, to further improve classification accuracy.
- **Background Confusion**: Misclassifications with the "Background" category indicate the need to refine augmentation techniques or increase the diversity of the training dataset to reduce noise and ambiguity.

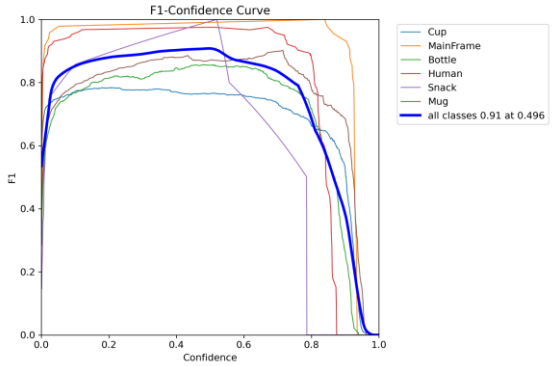


Figure 19 F1-Confidence Curve

F1-Confidence Curve Analysis

The F1-Confidence curve provides a detailed visualization of the model's performance across various confidence thresholds. The F1 score represents the harmonic mean of precision and recall, offering a balanced measure of detection accuracy.

Observations:

1. **Overall Performance**:
    - For all classes combined, the model achieved an optimal F1 score of **0.91** at a confidence threshold of **0.496**, demonstrating a strong balance between precision and recall at this point.
2. **Class-Specific Insights**:
    - **MainFrame and Human Classes**:
        - These classes consistently maintain higher F1 scores across a wide range of confidence thresholds.
        - The robust performance indicates reliable detection of restricted areas and human presence, even under challenging scenarios.
    - **Cup and Bottle Classes**:
        - These classes exhibit lower F1 scores at higher confidence thresholds, reflecting the misclassification issues noted in the confusion matrix.
        - This highlights the need for further augmentation or refinement in labeling and dataset diversity for these classes.
3. **Challenges and Improvements**:
    - **Cup**: The F1 score drops significantly at higher confidence levels, aligning with confusion matrix observations where cups were often misclassified as background.
    - **Bottle**: Similar trends indicate challenges in distinguishing bottles from background clutter or similar-shaped objects.

The F1-Confidence curve illustrates the model's strong performance across major classes like MainFrame and Human, ensuring reliable detection in critical areas. However, it also emphasizes areas for improvement, particularly in Cup and Bottle classifications. Adjustments to data augmentation and fine-tuning confidence thresholds during inference can further enhance the model’s accuracy and generalization capabilities.

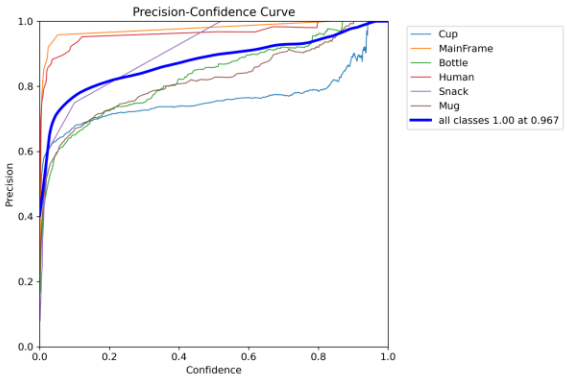


Figure 20 Precision-Confidence Curve

Precision-Confidence Curve Analysis

This curve visualizes how the model's precision—its ability to make correct predictions—varies with different confidence thresholds.

Key Observations:

1. **High Precision at Elevated Confidence Thresholds**:
    - At confidence thresholds approaching 1.0, the precision for all classes converges to nearly 100%, indicating the model's reliability in its high-confidence predictions.
    - This demonstrates that when the model is confident in its detections, its predictions are almost always accurate.
2. **Class-Specific Trends**:
    - **MainFrame and Snack**:
        - These classes exhibit consistently high precision across all confidence levels, suggesting the model is robust and less prone to false positives for these categories.
    - **Cup and Mug**:
        - Precision is lower at lower thresholds, reflecting challenges in distinguishing these classes due to similarities with other objects or the presence of false positives.
3. **Overall Model Performance**:

- For all classes combined, precision reaches 100% at a confidence threshold of 0.967. This indicates that the model is effective in minimizing errors when predictions are made with high certainty.

These insights highlight the model's strengths and areas needing improvement, particularly for smaller or visually similar objects. This analysis can guide further refinement efforts, such as enhanced data augmentation or additional training data for specific classes like "Cup" and "Mug."

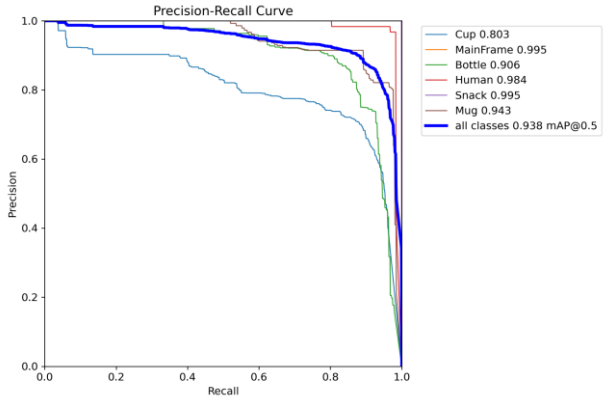


Figure 21 Precision-Recall Curve

Precision-Recall Curve Analysis

The Precision-Recall (PR) curve evaluates the balance between the model's precision (how many of its predictions are correct) and recall (its ability to detect all relevant objects) across various confidence thresholds. It provides valuable insights into the model's performance for each class.

Key Observations:

1. **Overall Performance**:

   - The model achieves a high **mean Average Precision (mAP@0.5)** of **93.83%**, indicating strong detection performance across all classes.
   - This result reflects the robustness of the model in balancing precision and recall while maintaining high detection accuracy.

2. **Class-Specific Analysis**:

    - **Bottle Class**:
        - A precision score of **0.906** demonstrates reliable detection for this class.
        - However, some decline in recall at higher confidence thresholds suggests a trade-off between missed detections and reduced false positives.
    - **Snack and MainFrame Classes**:
        - These classes exhibit near-perfect performance, with precision and recall curves tightly packed in the upper-right corner of the graph.
        - The model shows exceptional reliability in detecting these objects, even at higher confidence thresholds.
3. **Cup and Mug Classes**:

    - These classes show noticeable drops in recall at higher confidence thresholds, reflecting challenges in detecting smaller or more visually ambiguous objects.
    - The decline highlights areas where additional data augmentation or class-specific fine-tuning may improve performance.
4. **Recall Trends Across Confidence Thresholds**:

    - At lower confidence thresholds, recall for all classes approaches **1.0**, indicating the model's ability to detect most objects.
    - As the threshold increases, recall decreases for specific classes (e.g., **Cup**, **Bottle**) as the model prioritizes reducing false positives, becoming more conservative in its predictions.

The PR curve confirms the model's excellent generalization capabilities for most classes, particularly Snack, MainFrame, and Human. While the performance for Cup and Mug is relatively lower, targeted improvements such as data augmentation and focused training could enhance their detection accuracy. These findings provide actionable insights for further optimization of the object detection system.

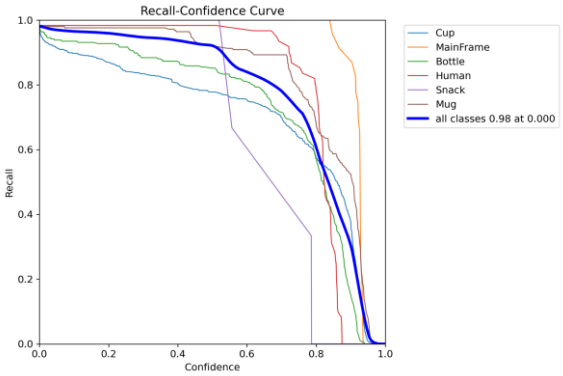


Figure 22 Recall-Confidence Curve

Recall-Confidence Curve Analysis

The Recall-Confidence curve provides insights into the model's ability to detect all relevant instances (recall) at varying confidence thresholds. It demonstrates how the model balances between capturing as many ground truth instances as possible and avoiding false negatives.

Key Observations:

1. **Overall Recall Performance**:
    - At lower confidence thresholds, recall for all classes approaches 1.0, indicating that the model is highly effective at capturing most objects when it is less conservative.
    - This behavior is expected, as lower thresholds allow the model to detect a larger number of instances, even at the cost of including some false positives.
2. **Class-Specific Trends**:
    - **MainFrame and Snack Classes**:
        - These classes maintain high recall across a wide range of thresholds, reflecting the model’s consistent ability to detect these objects, even under stricter conditions.
    - **Cup and Bottle Classes**:
        - Recall for these classes shows a noticeable drop at higher confidence thresholds. This indicates that the model becomes more selective, which leads to some true positives being missed.
    - **Mug and Human Classes**:

        - These classes demonstrate a balanced trend, with a gradual decline in recall as the confidence threshold increases.

3. **Trade-Off with Precision**:
    - The curve illustrates the trade-off between achieving high recall and avoiding false negatives. While recall is high at lower thresholds, it may be accompanied by a reduction in precision, as the model captures more irrelevant detections.
    - Conversely, higher thresholds improve precision but reduce recall, particularly for more visually ambiguous or smaller objects like **Cup** and **Bottle**.
4. **Implications for Real-World Applications**:
    - The high recall at lower thresholds indicates the model's suitability for scenarios where missing an object is critical, such as detecting restricted areas (MainFrame) or safety-critical objects.
    - For applications requiring higher confidence, thresholds can be adjusted to optimize recall for key classes while balancing precision.

The Recall-Confidence curve confirms that the model achieves excellent detection capabilities across all classes at lower thresholds, with specific strengths in detecting MainFrame, Snack, and Human objects. While classes like Cup and Bottle exhibit recall drops at stricter thresholds, targeted fine-tuning and additional data augmentation could further enhance their performance. This analysis highlights the importance of selecting appropriate thresholds based on application-specific priorities.

5.1.3 Testing Evaluation

The final step involved integrating the trained model with a live video feed to evaluate its real-time detection and tracking capabilities. During this phase, the model continuously detected and tracked cups in video frames, displaying the results in real-time. This allowed the team to assess both the detection accuracy and the speed at which the model processed the video. The real-time performance was monitored to ensure the system could process frames efficiently without compromising accuracy. The system demonstrated the ability to detect objects accurately while maintaining the real-time responsiveness required for the project's objectives. The modeling and simulation process allowed the team to fine-tune the detection system, optimizing it for accurate and efficient real-time object detection and tracking. The system's performance was validated

through both quantitative metrics and qualitative testing, ensuring its suitability for deployment in real-world scenarios.

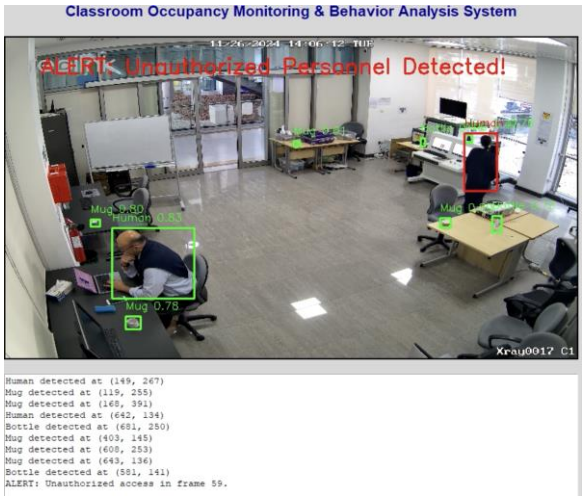


Figure 23 Successfully Detected and Alerted Human in Restricted Area

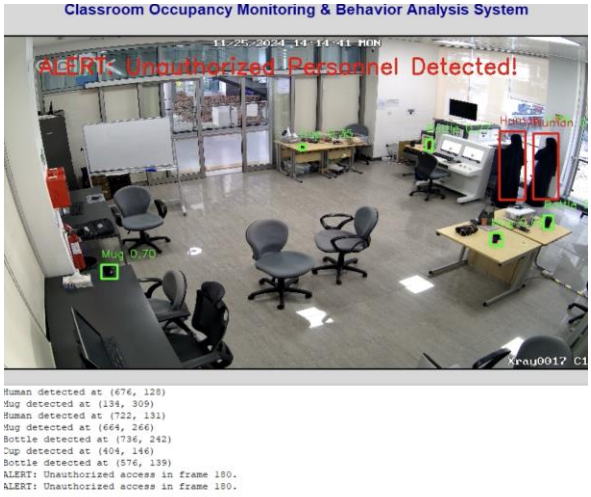


Figure 24 Successfully Identified Illegal Objects and 2 Humans in Restricted Area

Figure 25 Showcasing the Detection Capabilities of the System

The system incorporates a critical alert feature that captures and saves screenshots whenever unauthorized access is detected in the restricted "MainFrame" area. When the model identifies a person entering the restricted zone based on Intersection over Union (IoU) thresholds, it automatically saves a frame of the event as an image in the respective Alerts/VideoX directory. This feature enhances post-event evaluation by providing visual evidence of unauthorized activities, which can be used for security audits and further investigation. The systematic organization of these alert images ensures efficient access and review of critical events, bolstering the system's monitoring capabilities.

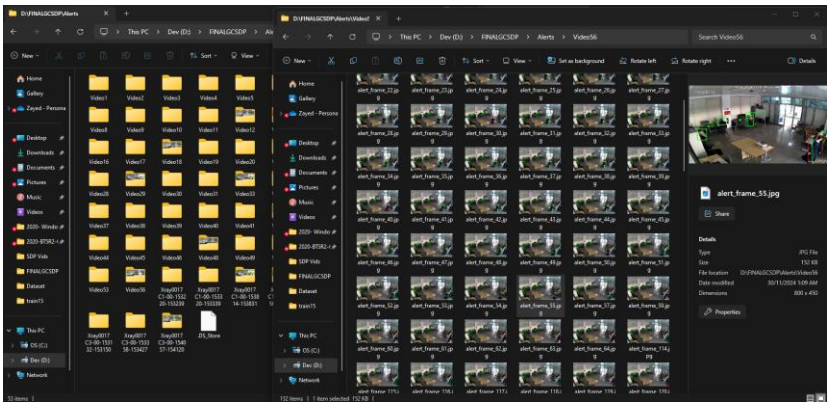
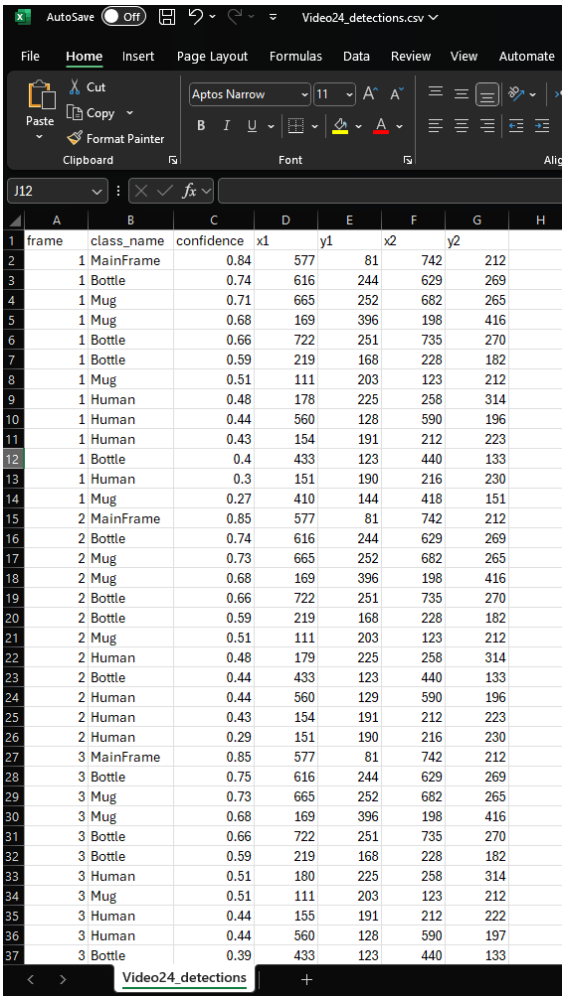


Figure 26 Alert Directory

For each processed video, the system generates a detailed CSV, as shown in the figure below, file that is automatically saved in the directory structure under runs/detect/videoX. This file provides a comprehensive, frame-by-frame summary of the detections made during real-time video analysis. The CSV contains essential information, such as the frame number, detected class names (e.g., "MainFrame," "Bottle," "Mug," etc.), confidence scores, and bounding box coordinates (x1, y1, x2, y2). These bounding box values represent the spatial location of the detected objects in the video frames. This feature allows for precise monitoring of object presence and positioning, enabling the team to analyze detection trends, verify alerts, and ensure accuracy. It serves as a valuable tool for post-analysis, debugging, and validation, offering insights into the model's performance across different classes and scenarios.



	A	B	C	D	E	F	G	H
1	frame	class_name	confidence	x1	y1	x2	y2	
2	1	MainFrame	0.84	577	81	742	212	
3	1	Bottle	0.74	616	244	629	269	
4	1	Mug	0.71	665	252	682	265	
5	1	Mug	0.68	169	396	198	416	
6	1	Bottle	0.66	722	251	735	270	
7	1	Bottle	0.59	219	168	228	182	
8	1	Mug	0.51	111	203	123	212	
9	1	Human	0.48	178	225	258	314	
10	1	Human	0.44	560	128	590	196	
11	1	Human	0.43	154	191	212	223	
12	1	Bottle	0.4	433	123	440	133	
13	1	Human	0.3	151	190	216	230	
14	1	Mug	0.27	410	144	418	151	
15	2	MainFrame	0.85	577	81	742	212	
16	2	Bottle	0.74	616	244	629	269	
17	2	Mug	0.73	665	252	682	265	
18	2	Mug	0.68	169	396	198	416	
19	2	Bottle	0.66	722	251	735	270	
20	2	Bottle	0.59	219	168	228	182	
21	2	Mug	0.51	111	203	123	212	
22	2	Human	0.48	179	225	258	314	
23	2	Bottle	0.44	433	123	440	133	
24	2	Human	0.44	560	129	590	196	
25	2	Human	0.43	154	191	212	223	
26	2	Human	0.29	151	190	216	230	
27	3	MainFrame	0.85	577	81	742	212	
28	3	Bottle	0.75	616	244	629	269	
29	3	Mug	0.73	665	252	682	265	
30	3	Mug	0.68	169	396	198	416	
31	3	Bottle	0.66	722	251	735	270	
32	3	Bottle	0.59	219	168	228	182	
33	3	Human	0.51	180	225	258	314	
34	3	Mug	0.51	111	203	123	212	
35	3	Human	0.44	155	191	212	222	
36	3	Human	0.44	560	128	590	197	
37	3	Bottle	0.39	433	123	440	133	

Figure 27 Output CSV File that Shows Confidence Rate and Coordinates of Identified Objects

Accuracy Evaluation:

The model achieved an impressive performance in its evaluation metrics, with an mAP@0.5 of 93.83% and an mAP@0.5:0.95 of 69.00%. These results highlight the model's capability to accurately detect and classify objects across various scenarios, with a strong emphasis on precision at higher thresholds. The overall accuracy was reported at 76.00%, indicating reliable predictions in real-world conditions. Each class demonstrated robust detection performance, with particularly high results for "MainFrame" and "Mug," achieving near-perfect mAP scores. The evaluation also underscored areas for improvement, such as further refining detection for smaller objects like "Cup" and "Bottle," to enhance the model's adaptability across diverse settings. The balance of precision, recall, and inference speed highlights the model's suitability for deployment in real-time object detection tasks.

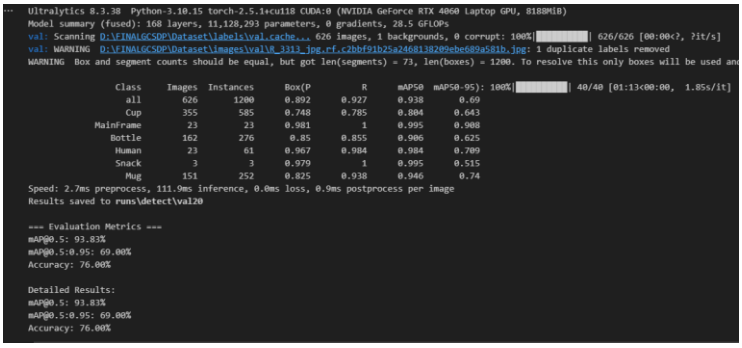


Figure 28 Model Accuracy Evaluation

5.2 Failure cases Analysis

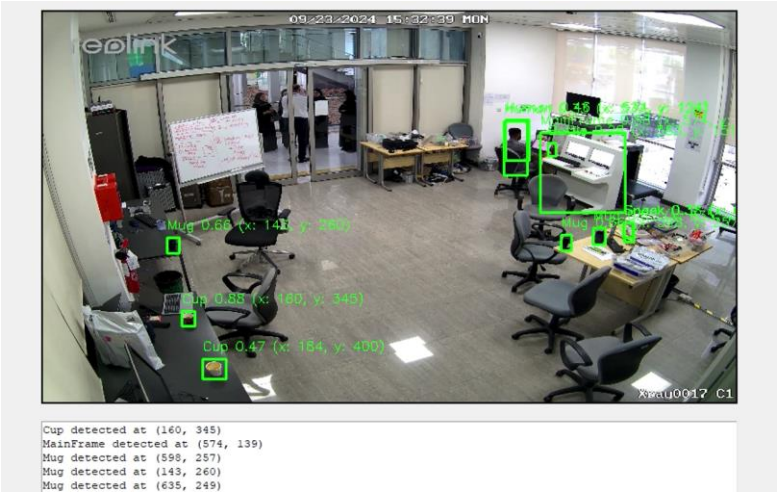


Figure 29 Failure to Alert Human in Restricted Area

The system successfully detects the "MainFrame" within the designated area, which is defined as an illegal area. However, the system fails to trigger an alert regarding the unauthorized person entering this restricted zone. Additionally, the detection system should not display the "MainFrame" boundaries, as the system is designed for keeping the restricted area hidden from view.

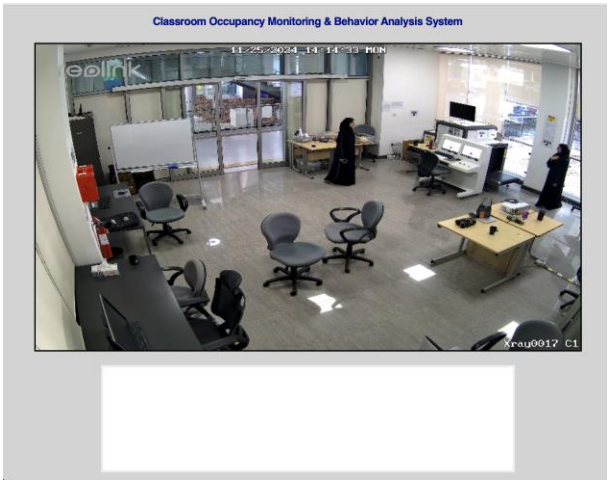


Figure 30 Failure to Detect Anything

The system has not detected any objects or persons in the illegal area. This shows that the space is currently empty, with no noticeable activities or events taking place.

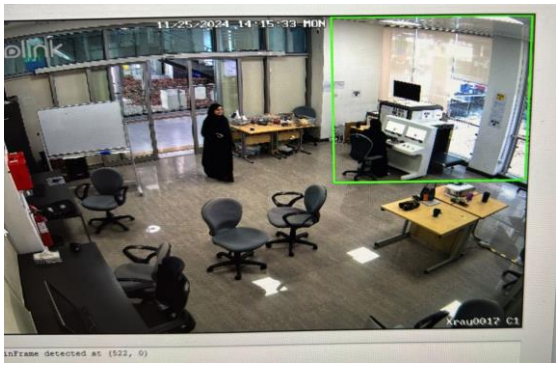


Figure 31 Only Detecting MainFrame

The system incorrectly detects only the "MainFrame," even though there is no person entering the illegal area to detect in the scene. This is a failure, as the system should not flag the "MainFrame" in the absence of any relevant activity or interaction. The detection is unnecessary and misleading.

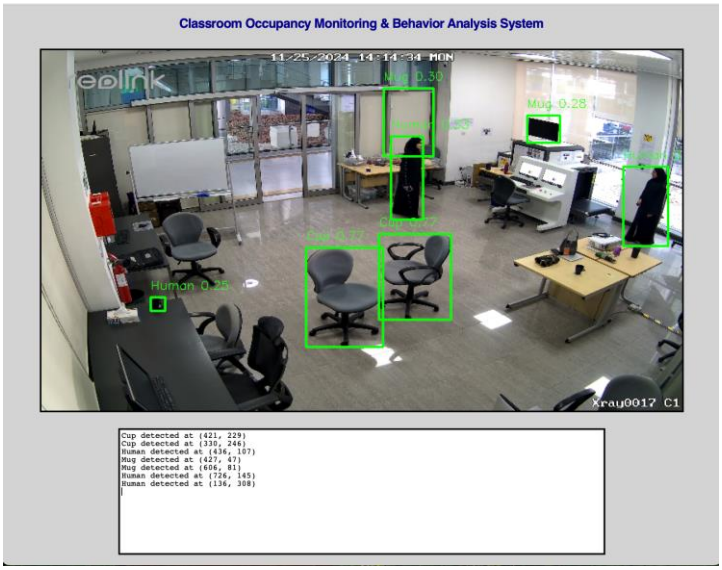


Figure 32 Incorrect Object Classification

The system shows incorrect labels. For example, a chair is mistakenly labeled as a "Cup," which is clearly wrong. Additionally, some "Human" labels appear in areas where there are no people, and other objects, like "Mug," are either mislabeled or detected with low confidence.

Chapter 6 Conclusion

6.1 Summary of work done

The project focused on extensively researching object detection techniques and exploring methodologies to accurately identify and classify items within video feeds. A comprehensive dataset of scanned images featuring various objects of interest was collected, annotated, and prepared for training the YOLOv8 model. These annotations included bounding box coordinates and confidence scores, forming the foundation for developing a robust detection system.

For system implementation, the project team worked on building a prototype capable of analyzing live video feeds to identify specific objects, such as cups or other items, and assess their presence in real-time. In addition, the system is also configured by logging the events it detects and saving them in a structured format, i.e., .csv files, for event analysis and record keeping, thus making it more useful in monitoring reports as well.

6.2 Conclusion Statement

This project demonstrates the effective integration of artificial intelligence and machine learning to enhance object detection and monitoring systems. Through collaborative teamwork, the team successfully curated and processed a comprehensive dataset, enabling the model to accurately identify, track, and classify objects in real-time scenarios. The collective efforts and shared expertise of the team were instrumental in overcoming challenges and achieving a significant milestone—developing a reliable system that enhances safety and efficiency in automated surveillance applications.

6.3 Critical appraisal of work done

This project achieved substantial progress in developing a real-time object detection system using YOLOv8. The team successfully implemented a pipeline that combined data preparation, feature engineering, and model training, producing a system capable of detecting and tracking

objects with promising accuracy. Collaborative teamwork played a vital role in overcoming challenges, with every member contributing to research, development, and testing efforts.

Despite these achievements, several obstacles were encountered during the process. One major challenge was the need for additional training on a larger dataset to improve the model's generalization capabilities. This was particularly important as some objects were occasionally not detected in the testing set, highlighting gaps in the training data. Another limitation was the slow runtime of the system, which affected its efficiency during real-time testing. Addressing this would require further optimization of the model and exploration of lightweight architectures better suited for hardware constraints.

6.3.1 Challenges and Solutions

Throughout the development process, several challenges emerged, and specific solutions were implemented to address them effectively:

Video Feed Quality: Capturing clear video recording in low-light conditions was a significant challenge. This issue was tackled by applying image preprocessing techniques, which enhanced the clarity of the video feeds and ensured reliable system performance, even in suboptimal lighting.

Model Accuracy: In the early stages, the system sometimes misclassified detection and alerts , leading to inconsistent results. To enhance accuracy, the models were adjusted and trained further using additional and varied datasets, enabling the system to perform more reliably across various scenarios.

Scalability: The system needed to support multiple cameras and process large amounts of video data, which initially led to challenges with processing capacity. To resolve this, the system's design was restructured and optimized, enabling it to handle higher data loads effectively and operate smoothly as it scaled up.

6.4 Next Steps

Looking forward, the team is planning to enhance the system by adding features that will improve its functionality and versatility. One key area of development is behavioral analysis, where the system will be trained to detect specific actions, such as eating or drinking, within real-time video streams. This will involve collecting additional datasets and fine-tuning the model to

recognize these behaviors accurately, and further training will be conducted to ensure the model can generalize effectively to new data and scenarios

Additionally, the team intends to implement occupancy detection to monitor the number of people present in a given space. This feature will help track classroom occupancy and ensure the system is capable of counting individuals within specific areas.

References

[1] T. R. Ruggles and J. M. LeBlanc, "Behavior Analysis Procedures in Classroom Teaching," in *International Handbook of Behavior Modification and Therapy*, Springer, Boston, MA, 1985. [Online]. Available: https://doi.org/10.1007/978-1-4615-7278-7_11.

[2] J. H. Lim, E. Y. Teh, M. H. Geh, and C. H. Lim, "Automated classroom monitoring with connected visioning system," in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Kuala Lumpur, Malaysia, 2017, pp. 386–393. doi: 10.1109/APSIPA.2017.8282063.

[3] Y. Nureni and O. Asiribo, "Networking CCTV cameras & Passive Infra-Red sensors for E-classroom monitoring system: Proactive approach to quality assurance in education system," ResearchGate, Jul. 2017. [Online]. Available: https://www.researchgate.net/publication/318636912.

[4] X. Zhao, G. Wang, Z. He, and H. Jiang, "A survey of moving object detection methods: A practical perspective," *Neurocomputing*, vol. 503, pp. 28–48, 2022. [Online]. Available: https://doi.org/10.1016/j.neucom.2022.06.104.

[5] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1680–1716, 2023. doi: https://doi.org/10.3390/make5040083.

[6] J. Pedro, "Detailed Explanation of YOLOv8 Architecture: Part 1," *Medium*, Dec. 4, 2023. [Online]. Available: https://medium.com/@juanpedro.bc22/detailed-explanation-of-yolov8-architecture-part-1-6da9296b954e. [Accessed: Nov. 1, 2024].

[7] V. Shehu and A. Dika, "Using real time computer vision algorithms in automatic attendance management systems," in *Proceedings of the ITI 2010, 32nd International Conference on Information Technology Interfaces*, Cavtat, Croatia, 2010, pp. 397–402.

[8] A. S. Pillai, "Student Engagement Detection in Classrooms through Computer Vision and Deep Learning: A Novel Approach Using YOLOv4," *Sage Science Review of Educational Technology*, vol. 5, no. 1, pp. 87–97, Aug. 2022. [Online]. Available: https://journals.sagescience.org/index.php/ssret/article/view/144.

[9] W. Zhang, J. Calautit, P. W. Tien, Y. Wu, and S. Wei, "Deep Learning Models for Vision-Based Occupancy Detection in High Occupancy Buildings," *Journal of Building Engineering*, 2024, Article ID 111355. [Online]. Available: https://doi.org/10.1016/j.jobe.2024.111355.

[10] "GPU Acceleration with PyTorch," *Toxigon*. [Online]. Available: https://toxigon.com/gpu-acceleration-pytorch. [Accessed: Nov. 14, 2024].

[11] S. Dewi, Y. Mahendra, and I. R. Kusuma, "Implementation of PyTorch Framework in Building Deep Learning Models," *Journal of Physics: Conference Series*, vol. 1693, no. 1, p. 012100, 2020. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1693/1/012100/meta. [Accessed: Nov. 14, 2024].

[12] G. Nitschke and L. Taylor, "Improving Deep Learning with Generic Data Augmentation," in *Proceedings of IEEE Symposium Series on Computational Intelligence (IEEE SSCI 2018)*, Bengaluru, India, 2018, pp. 1–8.

[13] "An Introduction to YOLOv8," *RevComm Tech Blog*. [Online]. Available: https://tech.revcomm.co.jp/yolov8-introduction-en. [Accessed: Nov. 16, 2024].

[14] S. Jiang, H. Qin, B. Zhang, and J. Zheng, "Optimized Loss Functions for Object Detection: A Case Study on Nighttime Vehicle Detection," *arXiv preprint arXiv:2011.05523*, 2020. [Online]. Available: https://arxiv.org/abs/2011.05523.

[15] T. Kumar, A. Mileo, R. Brennan, and M. Bendechache, "Image Data Augmentation Approaches: A Comprehensive Survey and Future Directions," *arXiv preprint arXiv:2301.02830*, 2023. [Online]. Available: https://arxiv.org/abs/2301.02830.

[16] "Mean Average Precision (mAP) Explained," *DigitalOcean Community*. [Online]. Available: https://www.digitalocean.com/community/tutorials/mean-average-precision. [Accessed: Nov. 17, 2024].

[17] B. Carterette, "Precision and Recall," in *Encyclopedia of Database Systems*, Springer, 2009. [Online]. Available: https://link.springer.com/referenceworkentry/10.1007/978-0-387-39940-9_5050. [Accessed: Nov. 17, 2024].

[18] P. Henderson and V. Ferrari, "End-to-End Training of Object Class Detectors for Mean Average Precision," in *Computer Vision – ECCV 2016*, Springer, 2016, pp. 198–213. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-54193-8_13. [Accessed: Nov. 18, 2024].

[19] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, p. 6, 2020. [Online]. Available: https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-019-6413-7. [Accessed: Nov. 18, 2024].

Appendices

Appendix A: Source code

```
# Data Augmentation

import cv2

import os

import imgaug.augmenters as iaa

# Configuration

video_dir = r"D:\Dataset Final\Yasmine"  # Directory containing videos

output_dir = r"D:\Dataset Final\Y output"  # Output directory for frames

num_frames_to_extract = 18  # Number of frames to extract

num_augmentations = 1  # Number of augmented frames per original frame

# Ensure output directory exists

if not os.path.exists(output_dir):

    os.makedirs(output_dir)

# Data augmentation sequence

augmenters = iaa.SomeOf((1, 3), [

    iaa.Fliplr(0.5),  # Horizontal flip

    iaa.Affine(rotate=(-15, 15)),  # Random rotation

    iaa.Multiply((0.8, 1.2)),  # Brightness adjustment

    iaa.AdditiveGaussianNoise(scale=(10, 30)),  # Add noise

    iaa.GaussianBlur(sigma=(0.0, 3.0)),  # Blur

])
```

```
# Process all video files in the directory
video_files = [f for f in os.listdir(video_dir) if f.endswith('.mp4')]
frame_id = 0  # Global frame counter to ensure unique filenames
for video_file in video_files:
    video_path = os.path.join(video_dir, video_file)
    cap = cv2.VideoCapture(video_path)
    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
    interval = max(total_frames // num_frames_to_extract, 1)
    print(f"Processing: {video_file}, Total frames: {total_frames}, Interval: {interval}")
    frame_count = 0
    extracted_frames = 0
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret or extracted_frames >= num_frames_to_extract:
            break
        # Extract frames at regular intervals
        if frame_count % interval == 0:
            # Save original frame with a unique name
            original_frame_path = os.path.join(output_dir, f"frame_{frame_id}.jpg")
            cv2.imwrite(original_frame_path, frame)
            # Generate augmented frames
            for i in range(num_augmentations):
```

```
            augmented_frame = augmenters(image=frame)
            augmented_frame_path = os.path.join(output_dir, f"frame_{frame_id}_aug_{i}.jpg")
            cv2.imwrite(augmented_frame_path, augmented_frame)
        extracted_frames += 1
        frame_id += 1
    frame_count += 1
cap.release()
print(f"{extracted_frames} frames (with augmentations) extracted from {video_file} and saved to {output_dir}")
print(f"All videos processed. Frames saved to {output_dir}")
```

```
# Model Train
import pandas as pd
from ultralytics import YOLO
# Load YOLO model
model = YOLO('yolov8s.pt')
model.data = r"D:\FINALGCSDP\Dataset\data.yaml"
print("Dataset configuration:", model.data)
# Train the model
model.train(
    data= r"D:\FINALGCSDP\Dataset\data.yaml",
    epochs=50,
```

```
    imgsz=640,
    batch=21,
    device = '0', # Using the GPU, If you get an error here, check how to use GPU with PyTorch
)
# Save the retrained model
model.save(r"D:\FINALGCSDP\yolov8s_retrained_NEW.pt")
print("Model saved successfully!")
# Start validation
print("Starting validation...")
metrics = model.val(
    data= r"D:\FINALGCSDP\Dataset\data.yaml",
    batch=3,
    imgsz=640
)
# Inspect metrics object
print("Metrics object content:")
print(metrics)
# Debug: Check available attributes in metrics
print("Available attributes in metrics:")
print(dir(metrics))
# Extract metrics
try:
    metrics_dict = {
```

```
        "precision": float(metrics.box[0]) if hasattr(metrics, 'box') else None,
        "recall": float(metrics.box[1]) if hasattr(metrics, 'box') else None,
        "mAP@50": float(metrics.box[2]) if hasattr(metrics, 'box') else None,
        "mAP@50-95": float(metrics.box[3]) if hasattr(metrics, 'box') else None
    }
except Exception as e:
    print(f"Error extracting metrics: {e}")
    metrics_dict = {
        "precision": None,
        "recall": None,
        "mAP@50": None,
        "mAP@50-95": None
    }
# Debug: Check extracted metrics
print("Extracted metrics:")
print(metrics_dict)
# Save metrics to CSV
csv_file_path = r"D:\FINALGCSDP\ValidationMetrics.csv"
try:
    metrics_df = pd.DataFrame(list(metrics_dict.items()), columns=["Metric", "Value"])
    metrics_df.to_csv(csv_file_path, index=False)
    print(f"Validation metrics saved to {csv_file_path}")
except Exception as e:
```

```
    print(f"Error while saving metrics to CSV: {e}")


# interface and object detection
import cv2
import os
import tkinter as tk
from tkinter import Label, Text
from PIL import Image, ImageTk
from ultralytics import YOLO



# Load YOLO model
model = YOLO(r"D:\FINALGCSDP\runs\detect\train15\weights\best.pt")  # Adjust path
model.conf = 0.1  # Confidence threshold
# Directory containing videos
video_directory = r"D:\Dataset Final\Testing videos"
video_files = [os.path.join(video_directory, f) for f in os.listdir(video_directory) if f.endswith(('.mp4', '.avi'))]
# Alert directory
alert_dir =  r"D:\FINALGCSDP\Alerts"
os.makedirs(alert_dir, exist_ok=True)
# Initialize Tkinter interface
window = tk.Tk()
window.title("Classroom Occupancy Monitoring & Behavior Analysis System")
```

```
window.geometry("900x700")

window.configure(bg="lightgrey")

# Title Label

title_label = Label(

  window,

  text="Classroom Occupancy Monitoring & Behavior Analysis System",

  font=("Helvetica", 16, "bold"),

  bg="lightgrey",

  fg="darkblue"

)

title_label.pack(pady=10)

 # Video panel

video_label = Label(window, bg="black")

video_label.pack(pady=10)

# Text area for alerts

alert_text = Text(window, height=15, width=100, font=("Courier", 10), wrap="word", bg="white",
fg="black")

alert_text.pack(pady=10)

# Function to process a single video

def process_video(video_path):

    video_name = os.path.splitext(os.path.basename(video_path))[0]

    alert_text.insert(tk.END, f"Processing video: {video_path}\n")

    window.update()
```

```
# Create subdirectories for each video
runs_dir = os.path.join("runs", "detect", video_name)
os.makedirs(runs_dir, exist_ok=True)
alerts_dir = os.path.join(alert_dir, video_name)
os.makedirs(alerts_dir, exist_ok=True)
# Initialize CSV file
csv_path = os.path.join(runs_dir, f"{video_name}_detections.csv")
with open(csv_path, 'w') as csv_file:
    csv_file.write("frame,class_name,confidence,x1,y1,x2,y2\n")
cap = cv2.VideoCapture(video_path)
if not cap.isOpened():
    alert_text.insert(tk.END, f"Error: Could not open video {video_path}\n")
    window.update()
    return
def update_frame():
    ret, frame = cap.read()
    if not ret:
        cap.release()
        alert_text.insert(tk.END, f"Finished processing video: {video_name}\n")
        window.update()
        next_video()
        return
    # Resize frame for display
```

```
frame = cv2.resize(frame, (800, 450))
# Perform object detection
results = model.predict(
    source=frame,
    save=True,
    project=runs_dir,
    name="predict",
    exist_ok=True
)
alerts = []
human_boxes = []
sensitive_area_boxes = []


# Save detections to CSV and process results
with open(csv_path, 'a') as csv_file:
    for box in results[0].boxes:
        class_id = int(box.cls[0])
        label = model.names[class_id]
        x1, y1, x2, y2 = map(int, box.xyxy[0])
        confidence = box.conf[0]
        frame_number = int(cap.get(cv2.CAP_PROP_POS_FRAMES))
        # Write to CSV
        csv_file.write(f"{frame_number},{label},{confidence:.2f},{x1},{y1},{x2},{y2}\n")
```

```
# Handle MainFrame (restricted area)
if label == "MainFrame":
    # Increase size of MainFrame bounding box by 50%
    width = x2 - x1
    height = y2 - y1
    new_width = int(width * 1.5)
    new_height = int(height * 1.5)
    center_x = x1 + width // 2
    center_y = y1 + height // 2

    x1 = max(0, center_x - new_width // 2)
    y1 = max(0, center_y - new_height // 2)
    x2 = min(frame.shape[1], center_x + new_width // 2)
    y2 = min(frame.shape[0], center_y + new_height // 2)
    sensitive_area_boxes.append((x1, y1, x2, y2))
    continue
# Draw bounding box
cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
text = f"{label} {confidence:.2f}"
cv2.putText(frame, text, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 1)
# Handle Humans
if label == "Human":
    human_boxes.append((x1, y1, x2, y2))
```

```
            alerts.append(f"{label} detected at ({x1}, {y1})")
        # Check for unauthorized access
        alert_triggered = False
        for human_box in human_boxes:
            for sensitive_area_box in sensitive_area_boxes:
                if calculate_iou(human_box, sensitive_area_box) > 0:  # Any overlap triggers alert
                    alert_triggered = True
                    frame_number = int(cap.get(cv2.CAP_PROP_POS_FRAMES))
                    alert_path = os.path.join(alerts_dir, f"alert_frame_{frame_number}.jpg")
                    cv2.imwrite(alert_path, frame)
                    # Draw red bounding box and alert text
                    cv2.rectangle(frame, (human_box[0], human_box[1]), (human_box[2], human_box[3]), (0, 0,
255), 2)
                    cv2.putText(frame, "Human", (human_box[0], human_box[1] - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)
                    alerts.append(f"ALERT: Unauthorized access in frame {frame_number}.")
        if alert_triggered:
            alert_text.insert(tk.END, "ALERT: Unauthorized Personnel Detected!\n")
            window.update()
            # Display alert text on the frame in red
            cv2.putText(frame, "ALERT: Unauthorized Personnel Detected!", (50, 50),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
        # Update alert box in the interface
        alert_text.delete(1.0, tk.END)
```

```
        for alert in alerts:
            alert_text.insert(tk.END, alert + "\n")

        # Convert frame for display
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        img = Image.fromarray(frame_rgb)
        img_tk = ImageTk.PhotoImage(image=img)
        video_label.imgtk = img_tk
        video_label.configure(image=img_tk)

        # Schedule next frame update
        window.after(10, update_frame)

    update_frame()

# Global video index
video_index = 0

# Function to start the next video
def next_video():
    global video_index
    if video_index < len(video_files):
        process_video(video_files[video_index])
```

```
            video_index += 1
        else:
            alert_text.insert(tk.END, "All videos processed.\n")
            window.update()

# Function to calculate IoU
def calculate_iou(boxA, boxB):
    xA = max(boxA[0], boxB[0])
    yA = max(boxA[1], boxB[1])
    xB = min(boxA[2], boxB[2])
    yB = min(boxA[3], boxB[3])
    inter_area = max(0, xB - xA) * max(0, yB - yA)
    boxA_area = (boxA[2] - boxA[0]) * (boxA[3] - boxA[1])
    boxB_area = (boxB[2] - boxB[0]) * (boxB[3] - boxB[1])
    return inter_area / float(boxA_area + boxB_area - inter_area)
# Start the first video
next_video()
# Run the Tkinter event loop
window.mainloop()
```

Appendix B: Logbook

Date	Event/Meeting	Details
10/09/2024	Project Assignment	Official assignment of the project "Classroom Occupancy Monitoring and Behavior Analysis System Using CCTV Cameras." Supervisors: Dr. Naoufel, Dr. Maregu Asefa, Dr. Muzammal Nasser.
16/09/2024	Project Assignment Mix-up	A delay occurred due to a mix-up with the project assignment, which was resolved after re-clarification.
18/09/2024	First Meeting with Dr. Maregu Asefa	Discussion with Dr. Maregu about the project's core objectives. Initial guidance was provided on what to focus on, including occupancy monitoring and behavior detection using CCTV cameras.
23/09/2024	First Meeting with Dr. Naoufel with Dr. Maregu	Detailed meeting with Dr. Naoufel, who explained the project deliverables, steps to follow, and expected outcomes. Video data collection from the lab's CCTV cameras was initiated with the help of Dr. Naoufel and Dr. Maregu.
24/09/2024	Meeting with Dr. Muhammad Nasser	Met with Dr. Muhammad Nasser to discuss technical challenges, including object detection and occupancy monitoring using video data. His insights helped refine the approach for handling the data and models.
25/09/2024 - 30/09/2024	Research and Understanding Phase	Each team member researched object detection models (YOLO), behavior monitoring techniques, fine-tuning methods, and real-time monitoring/reporting systems. This phase improved our understanding of how to implement the system effectively.
30/09/2024	Meeting with Dr. Maregu Asefa	Meeting with Dr. Maregu after the team had a better understanding of the project requirements. He provided resources on machine learning models to help with object and behavior detection.
01/10/2024	Initiation of Object Detection	Work began on object detection using the video data collected from the lab. Focus was on detecting prohibited items such as cups.
03/10/2024	Splitting Progress Report Sections	The team divided the progress report sections to manage the workload more effectively, allowing parallel work on technical documentation and model performance evaluations.
06/10/2024	Object Detection Errors	Implemented basic object detection, but errors occurred. The system detected only one object at a time, and there were misclassifications where

		objects were detected as people. The next step is to resolve these issues by refining the dataset and improving model accuracy.
09/10/2024	Person Detection with Errors	Started working on person detection using the YOLO model. However, errors persist where the system sometimes misclassifies objects as people. We will work on refining the model to reduce these false positives.
18/10/2024	Progress Report Submission	Our progress report was submitted and sent through email to Dr. Naoufel, Dr. Hasan, Dr. Muhammad Nasser, Dr. Maregu Habtie.
21/10/2024	Meeting with Dr. Naoufel	Our group discussed with Dr. Naoufel the next steps for the project, focusing on validating detection rates and implementing a "zone intrusion" feature. This would check if the center of mass is within a restricted area. Additionally, we plan to create an interface to raise alerts if someone enters an unauthorized zone or if a cup is detected. The team is now working on integrating these features into the system.
24/10/2024	Model Training using MakeSense.ai	We used MakeSense.ai for annotating three videos as the training set and the remaining videos for testing. This step aimed to improve the dataset quality and help the model differentiate between people and objects more effectively.
29/10/2024	Errors when unauthorized access when testing	Testing revealed that whenever anyone entered the restricted areas sometimes a red flag would not appear. This behavior highlighted the need to refine the dataset and retrain the model to ensure accurate detection of unauthorized access.
30/10/2024	Testing the model on different videos	Videos recorded from different angles in the lab were used to test the model's performance.
8/11/2024	Generating Confusion Matrix	We generated a confusion matrix to analyze the model's performance in detail. This allowed us to understand where errors were occurring (false positives and negatives) and plan for further improvements.
14/11/2024	Working on the interface with using Tikner	Started building the system interface using Tkinter. Encountered issues integrating Gradio due to compatibility errors.

14/11/2024	Writing SDP I final report	The team began drafting the final report for SDP I, consolidating progress, results, and challenges faced so far. Each member contributed to their respective sections, ensuring all details were documented clearly.
14/11/2024	Data collection	The team gathered additional data to enhance the model's training process, ensuring it could achieve higher accuracy and generalize better to unseen scenarios
20/11/2024	Data collection for testing phase	The team expanded the dataset by collecting additional data from RoboFlow to enhance the diversity of objects such as cups, mugs, and bottles.
25/11/2024	Meeting with Dr. Naoufel	Dr. Naoufel thoroughly reviewed our project report and provided detailed feedback on improving its clarity, coherence, and overall presentation. He emphasized the importance of refining the structure, enhancing the technical explanations
25/11/2024	SDP Presentation Preparation	Work began by splitting and creating slides for the SDP presentation, highlighting major achievements, challenges, and future plans.
1/12/2024	Final SDP I Report Submission	The team finalized and submitted the comprehensive SDP report, detailing the project's progress, challenges, and outcomes. This submission included technical documentation, model performance analysis, and a summary of the testing results.
5/12/2024	SDP presentation	The team delivered the final presentation, showcasing the project to supervisors and examiners. The presentation highlighted key achievements, demonstrated the system's functionality, and addressed questions about potential improvements and future applications.

Appendix C: Gantt Chart

Gantt Chart	Classroom Occupancy Monitoring and Behavior Analysis System															
	Fall 2024															
Tasks	September				October				November				December			
	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4
Research, discovery, and project selection																
Meeting with instructors																
Literature review and project sepcification																
Progress report																
Data Collection and Prepration																
Collect classroom occupancy and behavior datasets																
Preprocess raw video data for model training																
Annotate data for object detection and restricted area																
Machine Learning Model Development																
Select appropriate pre-trained models (YOLO)					YOLO v5			YOLO v8								
Train occupancy detection models using collected datasets																
Test models for accuracy and adjust hyperparameters.																
User Interface Development																
Design a user-friendly interface																
Backend Development for Data Management																
Frontend Development for Visualization and Alerts																
Reporting & Documentation																
SDP I Report												1st Dec2024				
SDP I Presentation													5th Dec 2024			