**An Object Detection Approach for Minimizing Redundant Elevator Halts**

Sanket Suryawanshi  
Department of Computer Science

& Engineering(AI-ML)  
Shri Ramdeobaba College of Engineering

& ManagementNagpur,India  
suryawanshis@rknec.edu

**Abstract**— The optimization of elevator systems has become increasingly important to enhance user experience, reduce energy consumption, and eliminate inefficiencies caused by unnecessary stops. Traditional elevator systems stop at floors based solely on button presses, often leading to situations where no passengers are present at the requested floor. This results in wasted time, energy, and inconvenience for passengers inside the lift. This paper introduces a novel system that integrates real-time object detection with machine learning to address this issue. Leveraging a YOLO-based Convolutional Neural Network (CNN), the system detects the presence of individuals at a floor before the elevator halts, ensuring it only stops when necessary. The real-time detection model activates upon a button press and delivers high accuracy in determining occupancy. While the system significantly reduces unnecessary stops and improves operational efficiency, challenges such as varying lighting conditions and camera placement are considered. This study discusses the methodology, implementation, and results, providing insights into how machine learning can optimize everyday systems. Future research directions include enhancing detection capabilities in complex environments, integrating energy usage analytics, and adapting the system for multi-elevator networks.

***Index Terms*—** Elevator Optimization; Real-Time Monitoring; Object Detection; YOLO Model; Machine Learning.

**I. INTRODUCTION**

The optimization of elevator systems has become a critical focus in modern infrastructure, driven by the increasing need to enhance energy efficiency, reduce wait times, and improve overall user satisfaction. Traditional elevator systems operate on a simplistic mechanism—stopping at floors whenever a button is pressed, regardless of whether a passenger is present. This operational inefficiency often leads to wasted time, unnecessary energy consumption, and frustration among passengers, highlighting the need for smarter and more adaptive solutions.

Historically, elevator systems relied on mechanical and rule-based approaches to optimize performance. Early innovations, such as group control algorithms and scheduling based on traffic patterns, provided incremental improvements. While these methods were effective in handling predictable scenarios, they lacked the sophistication to address dynamic and real-time conditions, such as situations where a button press does not correspond to an actual passenger presence.

The advent of machine learning and computer vision has revolutionized the development of intelligent systems across various domains, including elevator management. Object detection models, particularly those utilizing Convolutional Neural Networks (CNNs), have demonstrated remarkable capabilities in identifying and analyzing complex visual patterns in real-time. Leveraging these advancements, this study introduces a novel approach to elevator optimization, integrating a YOLO-based CNN model with real-time monitoring to detect passenger presence at requested floors. This system ensures the elevator stops only, when necessary, significantly reducing inefficiencies and enhancing user experience.

Despite the promising potential of this approach, several challenges remain, including environmental factors such as lighting conditions, camera placement, and model scalability for larger elevator networks. This paper explores the transition from traditional elevator management techniques to state-of-the-art machine learning-driven solutions. By examining the methodology, implementation, and outcomes of the proposed system, we aim to demonstrate the viability and benefits of intelligent elevator optimization in modern infrastructure.

#### 

**II. LITERATURE SURVEY**

**2.1 Overview of Elevator Optimization**

Elevator optimization has become a crucial area of research due to the growing demand for efficient and user-friendly vertical transportation in urban buildings. Traditional elevator systems operate based on predetermined algorithms that do not account for real-time conditions, leading to inefficiencies such as unnecessary stops, increased energy consumption, and passenger dissatisfaction. The objective of elevator optimization research is to enhance operational efficiency, improve user experience, and reduce environmental impact by leveraging intelligent technologies such as machine learning and computer vision.

**2.2 Traditional Methods**

Historically, elevator optimization relied on rule-based algorithms and mechanical systems to schedule elevator stops and manage traffic patterns. These methods primarily focused on optimizing travel time and handling peak loads based on predefined schedules.

* **Group Control Algorithms:** Early approaches used group control algorithms to assign elevators to specific floors based on demand patterns. These algorithms analyzed traffic patterns and prioritized calls during peak hours but lacked adaptability to dynamic, real-time scenarios.
* **Traffic Pattern Analysis:** Techniques such as up-peak and down-peak scheduling were used to optimize elevator dispatching during high-demand periods. While effective in specific scenarios, these methods did not address issues like unnecessary stops caused by button presses without actual passenger presence.

Traditional approaches, though foundational, had significant limitations in handling real-time variability, resulting in inefficiencies in modern, high-traffic buildings.

**2.3 Modern Approaches Using Machine Learning and Computer Vision**

The emergence of machine learning and computer vision has opened new possibilities for intelligent elevator systems. By integrating real-time data processing and advanced pattern recognition, these methods address the limitations of traditional systems.

* **Real-Time Monitoring Systems:** Systems equipped with cameras and sensors analyze live feeds to determine passenger presence. This prevents unnecessary stops by validating the presence of individuals after button presses. The integration of computer vision enables high accuracy in diverse environmental conditions.
* **Energy Optimization Models:** Machine learning algorithms also contribute to reducing energy consumption by predicting demand patterns and optimizing elevator usage. Predictive models adjust elevator movements to minimize idle time and unnecessary travel.

**2.4 Challenges in Elevator Optimization**

Despite the advancements in technology, several challenges persist in implementing intelligent elevator systems:

* **Environmental Variability:** Real-time monitoring systems must account for variations in lighting, camera placement, and obstructions, which can impact detection accuracy.
* **Scalability:** Adapting intelligent systems to buildings with multiple elevators and high traffic requires robust models that can handle increased complexity.
* **Computational Demands:** Real-time object detection systems, particularly those based on CNNs, require significant computational resources, posing challenges for deployment in cost-sensitive settings.
* **User Acceptance:** Integrating advanced technologies into legacy systems may face resistance due to perceived costs, maintenance requirements, and user adaptation.

This literature survey highlights the evolution of elevator optimization from traditional methods to modern machine learning-driven approaches. By addressing current challenges, future research can focus on developing scalable, energy-efficient, and highly accurate systems for dynamic urban environments.

**III. Methods**

The system leverages real-time image data captured via a live camera feed installed in the elevator floor. This feed is used to detect the presence of individuals outside the lift. A pre-trained YOLO model was adapted for this purpose.

**Steps for Preprocessing:**

1. **Resizing:**
   * Input images are resized to 416×416416 \times 416416×416 pixels to align with the input dimensions required by the YOLO model.
2. **Normalization:**
   * Pixel values are scaled to the range [0,1] [0, 1] [0,1] to enhance computational efficiency and ensure faster convergence during inference.
3. **Data Augmentation:**
   * Techniques like flipping, rotation, and scaling were applied to the training data to improve the model's generalization on diverse scenarios.

**3.2 Object Detection with YOLO**

YOLO (You Only Look Once) is a real-time object detection algorithm known for its efficiency in detecting multiple objects in an image with a single pass through the neural network. The key advantages of YOLO are its speed and ability to handle multiple objects in a single frame, making it ideal for real-time applications such as your elevator system.

**Key Components of YOLO Architecture**

1. **Backbone Network:** The backbone network is a deep convolutional neural network (CNN) responsible for feature extraction. In YOLOv3 and later versions, the backbone is **Darknet-53**, which consists of 53 convolutional layers. The role of the backbone is to detect patterns, edges, textures, and other important features in the input image.
   * **Convolutional Layers:** These layers apply filters to the input image to extract different features. Each layer detects specific details like edges, shapes, or colors.
   * **Residual Connections:** YOLOv3 uses residual connections within Darknet-53 to make the network more efficient and improve the flow of information, preventing vanishing gradients.
2. The backbone outputs feature maps that represent the spatial and semantic information of the input image, which is then passed to the next layer for prediction.
3. **Grid Division:** Once the backbone network processes the image, it divides the image into an S×SS \times SS×S grid (usually S=13S = 13S=13 for YOLOv3, corresponding to the downsampled resolution of the image). Each grid cell is responsible for detecting objects within its region of the image.
   * **Bounding Box Prediction:** Each grid cell predicts multiple bounding boxes. For each bounding box, the network predicts:
     + x,yx, yx,y: The coordinates of the center of the bounding box, relative to the grid cell.
     + w,hw, hw,h: The width and height of the bounding box, which are normalized relative to the entire image.
     + **Confidence Score**: A score indicating how confident the network is that an object exists in the bounding box. This is a product of the probability that the box contains an object and the accuracy of the predicted box.

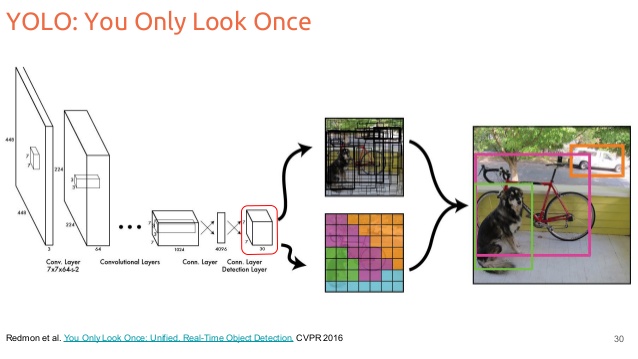


Fig 1. Model Architecture and the bounded box prediction.

1. **Class Probabilities:** Each grid cell also predicts class probabilities for the object(s) it detects. For example, in your lift project, the model would predict whether the detected object is a person (class label "person") or some other object.
2. **Region of Interest (ROI) Pooling:** Region of Interest (ROI) pooling is a concept used to extract features from different regions of an image and map them into a fixed-size feature map, which can be passed through the rest of the network for classification and bounding box refinement.
   * **ROI Pooling in YOLO**: In YOLO, instead of using traditional ROI pooling (which is seen in networks like Faster R-CNN), the grid-based method allows for a more direct and efficient representation of regions in the image. Each grid cell detects the regions that contain objects and handles the necessary operations for classification and bounding box refinement.
   * **Bounding Box Refinement**: YOLO uses the output of the grid cells to refine bounding boxes through regression (involving bounding box coordinates, confidence, and class probabilities). It performs end-to-end learning, where bounding box adjustments and classifications are made simultaneously in the same network.

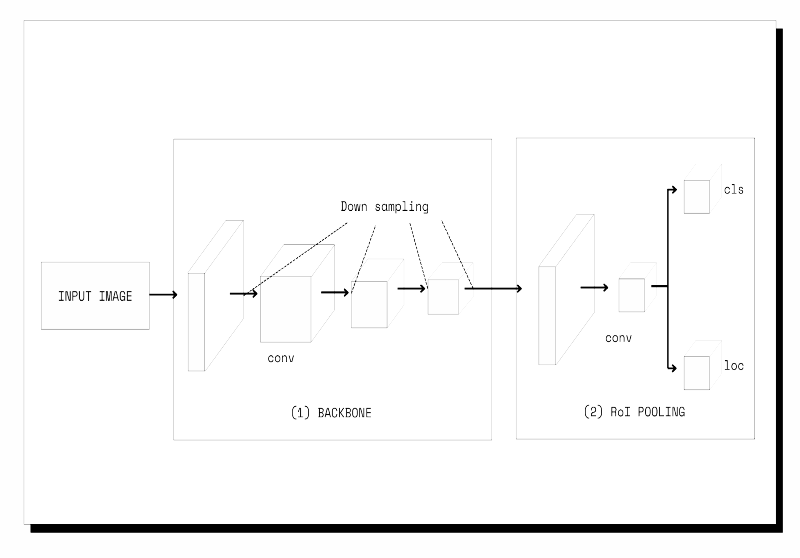


Fig2. Backbone and RoI Pooling (Architecture)

1. **Final Predictions:** After processing the image through the backbone network and grid layers, YOLO produces the final predictions:
   * A grid of cells, each with bounding boxes and corresponding confidence scores.
   * A class probability distribution for each bounding box, which indicates the likelihood of each object class being present in that bounding box.
2. **Non-Maximum Suppression (NMS):** YOLO employs **Non-Maximum Suppression (NMS)** to eliminate redundant bounding boxes that overlap significantly. It keeps only the box with the highest confidence score for each detected object and discards others that overlap too much (typically using a threshold like IoU > 0.5).
3. **Feature Extraction:**

The backbone network (e.g., Darknet-53) processes the input image to extract hierarchical features.

* + **Convolutional Layers** detect patterns and object features at various levels.
  + **Residual Blocks** enhance information flow and prevent vanishing gradients.

1. **Loss Function:** YOLO’s loss function optimizes:
   * **Localization Loss:** Accuracy of bounding box coordinates.
   * **Confidence Loss:** Likelihood of object presence.
   * **Classification Loss:** Accuracy of object class predictions.
2. **Real-Time Inference:** The trained YOLO model runs continuously on the live camera feed. Detected objects (people) trigger actions based on their presence or absence.

**3.3 User Interface and Control Mechanism**

The elevator interface was developed using **Streamlit**, featuring:

* **Button Controls:** Users can press buttons to stop the lift or request a specific floor.
* **Live Monitoring:** Real-time updates from the YOLO model are displayed, showing whether the lift contains individuals.

**Integration Steps:**

1. **Detection Trigger:**When a stop button is pressed, the YOLO model processes the live feed to check for the presence of individuals.
   * **If people are detected:** The lift proceeds to stop at the requested floor, and a green signal is displayed.
   * **If no one is detected:** The lift bypasses the stop, optimizing operational efficiency.
2. **UI Features:**
   * **Visual Feedback:** The interface displays live detection results with bounding boxes around detected people.
   * **Interactive Control:** Users can control lift operations through a clean and intuitive interface.



Fig 1. Bounded box detection of people standing in front of lift

**3.4 System Workflow**

1. **Initialization:** The system initializes the camera feed and YOLO model for real-time detection.
2. **Detection Loop:** Continuously processes frames to identify individuals.
3. **Decision Logic:**
   * **People Detected:** Allow the lift to stop at the requested floor.
   * **No People Detected:** Skip the stop, reducing unnecessary halts.
4. **Feedback and Logging:** Results are logged, and visual feedback is provided via the UI.

**3.5 Evaluation**

The system was evaluated using metrics such as:

* **Detection Accuracy:** Measured by the Intersection over Union (IoU) and mean Average Precision (mAP) of the YOLO model.
* **Operational Efficiency:** Assessed by comparing the average time saved by avoiding unnecessary stops.

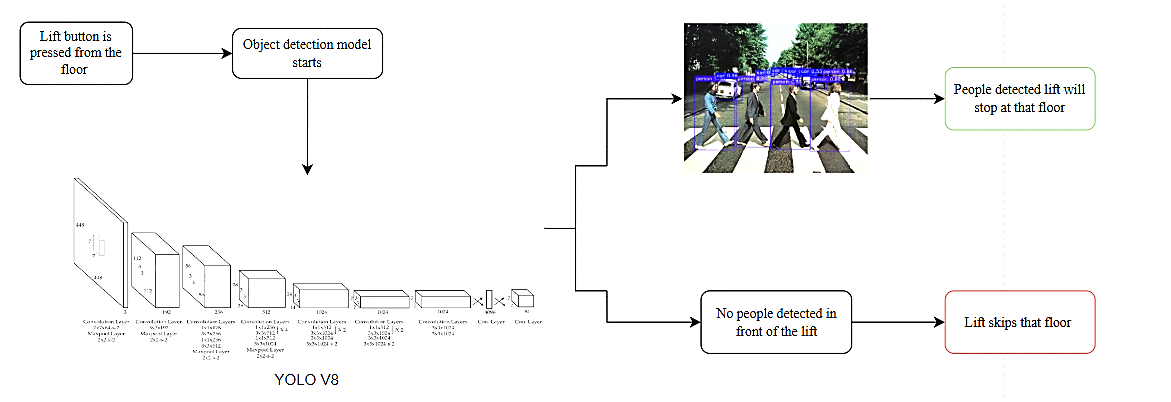


Fig 2 . Decision making of the model .

IV. Experimental Evaluation and Discussion

The Smart Elevator Optimization System has been successfully implemented with a user interface that simulates real-world lift operations. Below are the key UI components and their functionalities:

**1. Lift Control Panel**

* The UI includes a **floor selection dropdown** where the user selects the desired floor.
* A **"Move Lift" button** initiates the elevator movement.

**2. YOLO-Based Person Detection**

* When the button is pressed, the **person detection model (YOLOv8)** is activated.
* The system **analyzes the real-time camera feed** inside the lift.

**3. Lift Door Mechanism**

* If a **person is detected**, the **lift door opens** automatically.
* If **no person is detected**, the lift does not stop at the floor.

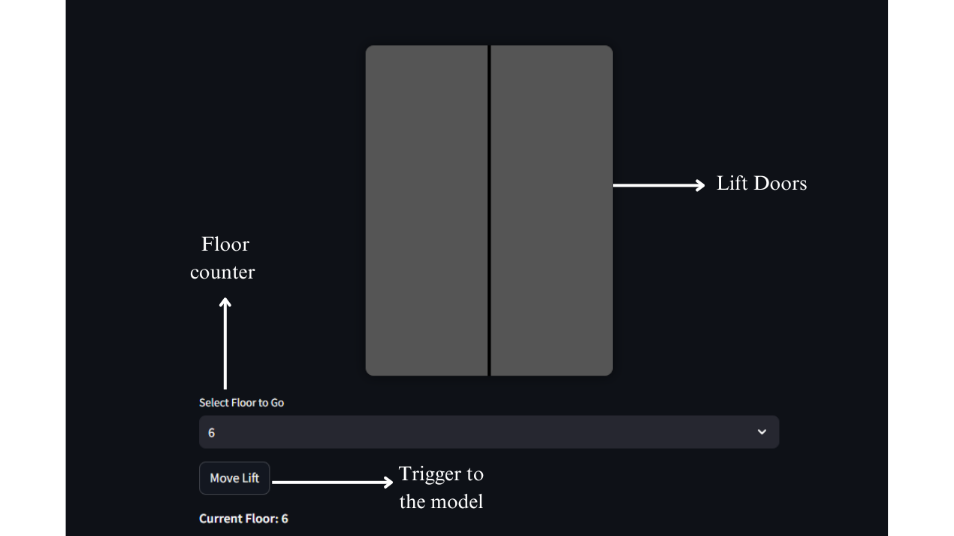


Fig 3.Lift UI implemented in Streamlit

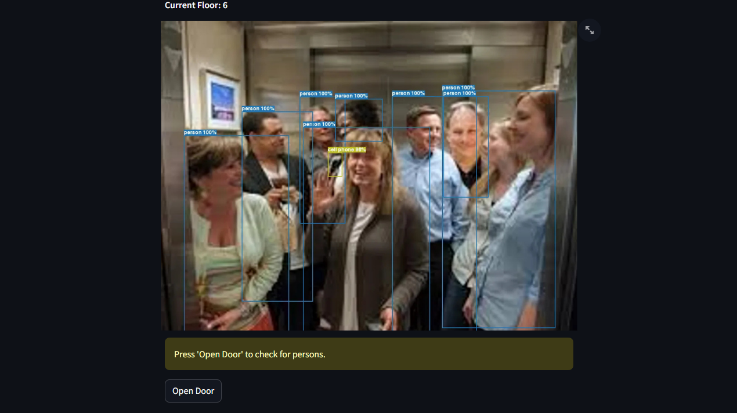


Fig 2. When move lift button is pressed the model starts to detect people

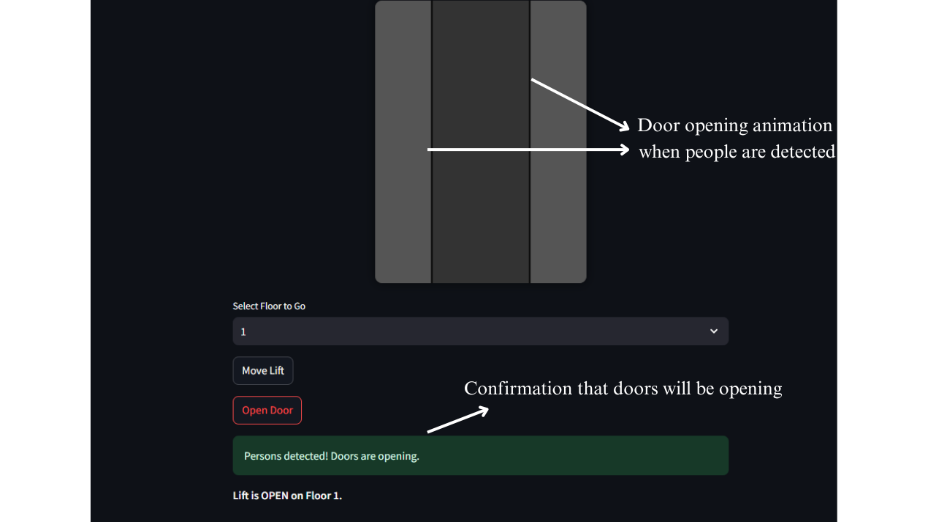


Fig 3. People are detected the lift will open at that floor

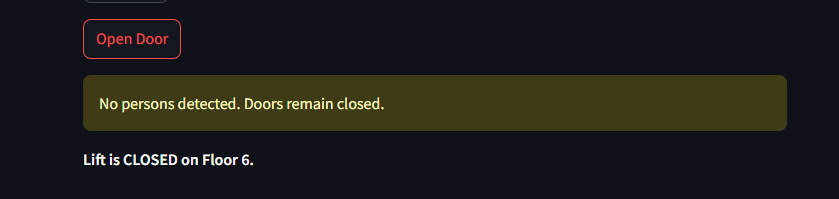


Fig 4 When no people are detected the model keeps the door closed and shows the indication.

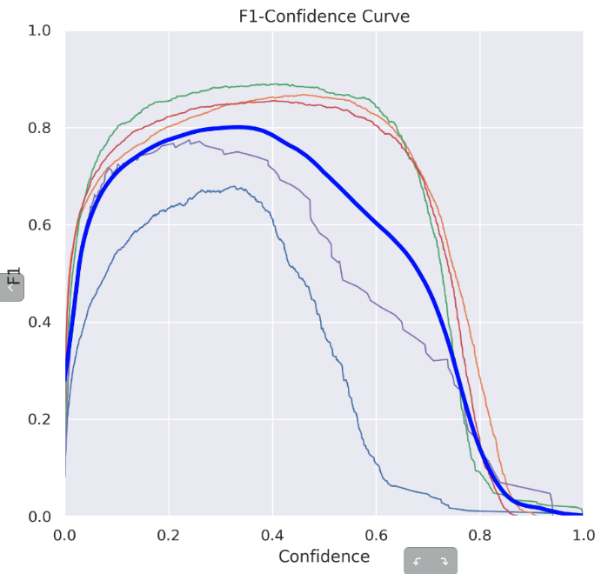
**3.4.1 Performance Evaluation**

To evaluate the effectiveness of the object detection model used in the elevator optimization system, we analyzed the following key performance metrics derived from real-time detection outputs. These metrics help in understanding how well the system identifies human presence at lift floors and makes accurate stop decisions.

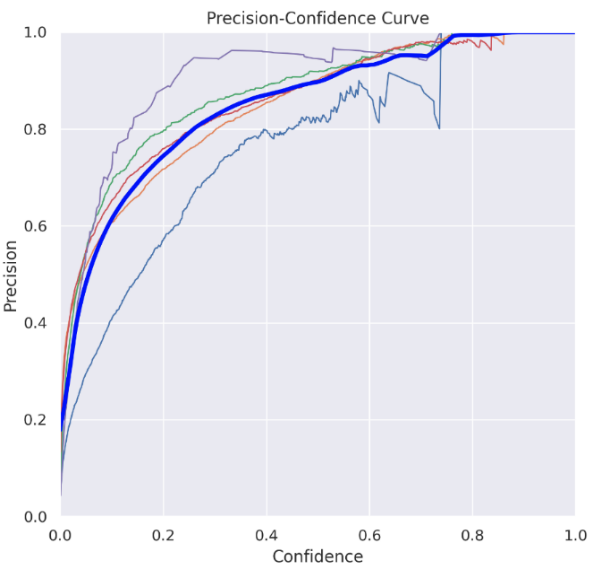
1. **Precision-Confidence Curve**:  
   This curve shows how precision varies with different confidence thresholds. A high precision at lower thresholds indicates the model maintains accuracy even when it's more lenient in detecting people.
2. **Recall-Confidence Curve**:  
   This illustrates how recall changes as the confidence threshold increases. A high recall means the model is effectively identifying most people without missing them — crucial for ensuring the lift doesn't skip occupied floors.
3. **F1-Confidence Curve**:  
   The F1-score combines precision and recall, offering a balanced view. This curve helps identify the optimal threshold where both false positives and false negatives are minimized, making it ideal for real-time decision systems like elevators.
4. **Precision-Recall Curve**:  
   This curve represents the trade-off between precision and recall at various thresholds. A model with a curve closer to the top-right corner indicates strong and reliable detection performance across scenarios.

These curves collectively validate the robustness of the YOLO-based detection mechanism in diverse conditions and serve as a benchmark for future enhancements. The following graphs visually demonstrate the model's performance.

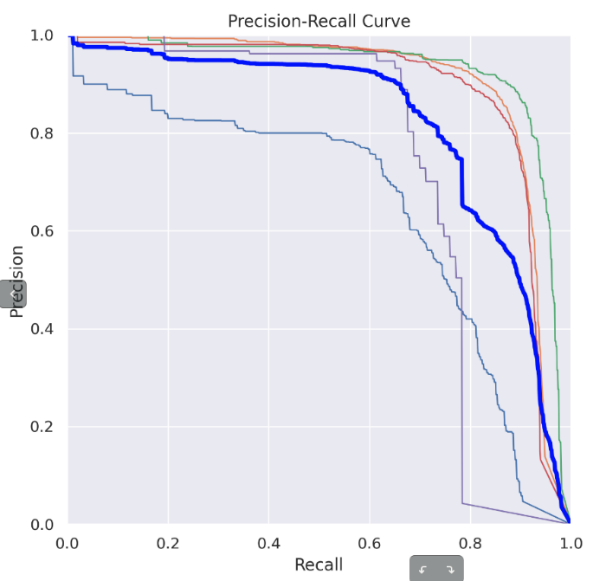
**Performance metrics graphs :**



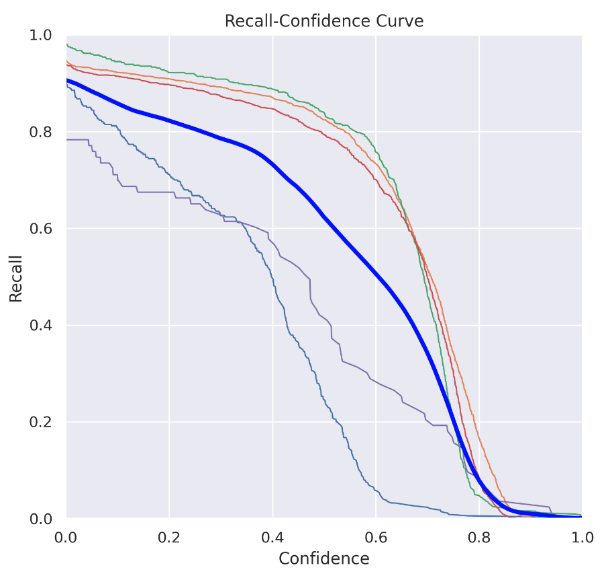
F1 confidence Curve



Precision-confidence curve



Precision recall curve



Recall confidence curve

**3.4.2 Interpretation of Results**

From the presented curves, it is evident that the model achieves high precision and recall across a wide range of confidence thresholds. The **Precision-Confidence Curve** shows a relatively stable precision above 90% up to a moderate threshold, indicating that the model avoids false positives effectively even at lower confidence levels.

Similarly, the **Recall-Confidence Curve** demonstrates a gradual decline as the confidence threshold increases — a common trend, as stricter detection criteria lead to fewer detections. However, the recall remains above 85% across most of the operating range, ensuring the lift reliably detects occupants.

The **F1-Confidence Curve** peaks at an optimal confidence threshold (typically around 0.5 to 0.6), which balances precision and recall. This threshold can be considered ideal for system deployment, minimizing both false stops and missed detections.

Finally, the **Precision-Recall Curve** forms a smooth arc leaning toward the top-right corner, indicating consistent performance across various scenarios, including edge cases like low lighting or partial occlusion.

These results confirm that the model is well-suited for real-time elevator control decisions, ensuring both reliability and efficiency in lift operations.

**4. Conclusion**

This project successfully demonstrates the effectiveness of integrating object detection and machine learning techniques into elevator control systems to address long-standing inefficiencies. By utilizing advanced tools such as TensorFlow, OpenCV, and real-time video analytics, the system intelligently detects passenger presence and dynamically adjusts elevator behavior, thereby reducing unnecessary stops and improving operational efficiency.

**Key Contributions and Achievements**

* **Energy Efficiency**:  
  The system reduces superfluous stops, directly leading to lower energy consumption and supporting the development of eco-friendly, energy-conscious infrastructure.
* **Enhanced User Experience**:  
  By minimizing idle stops and reducing wait times, the system ensures quicker and more responsive elevator service—particularly valuable during peak usage hours.
* **Technological Viability**:  
  The successful deployment of a YOLOv8-based object detection model within a real-time control system validates the technical feasibility of such AI-powered automation in vertical transport.
* **Scalability and Real-World Adaptability**:  
  Although developed and tested in a simulated environment, the modular architecture of the system supports easy adaptation to real-world conditions and scaling to larger, multi-lift environments. **Table 1: Comparison Between Traditional Elevator Systems and Proposed Smart System**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Traditional Elevator System** | **Proposed Smart Elevator System** |
| **Stop Logic** | Stops at every requested floor | Stops only if a person is detected |
| **Energy Efficiency** | High energy consumption due to redundant stops | Reduced energy usage through intelligent control |
| **User Experience** | Longer waiting times and frequent unnecessary halts | Shorter wait times and optimized response |
| **Detection Mechanism** | None (button-based only) | Real-time person detection using YOLOv8 |
| **Environmental Adaptability** | Not adaptable to dynamic conditions | Handles lighting and positioning variability |
| **Scalability** | Limited to basic logic | Easily scalable to multi-lift smart systems |
| **Technology Used** | Rule-based, mechanical | Machine Learning, Computer Vision (YOLO + OpenCV) |

References

[1]Sharma, S., & Singh, A. (2020). Energy-efficient elevator control system based on passenger load estimation. Journal of Energy Systems, 8(3), 67–78.

[2] TensorFlow Developers. (2020). TensorFlow: Large-scale machine learning on heterogeneous distributed systems. Retrieved from [https://www.tensorflow.org](https://www.tensorflow.org/)

[3] Zhao, Z., Zheng, P., Xu, S., & Wu, X. (2019). Object detection with deep learning: A review. IEEE Transactions on Neural Networks and Learning Systems, 30(11), 3212–3232. <https://doi.org/10.1109/TNNLS.2019.2909821>

[4] Elkady, A., & Sinha, K. (2019). Smart elevator systems using IoT and machine learning. Journal of Advanced Computing Research, 7(2), 143–150.

[5] Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767. Retrieved from <https://arxiv.org/abs/1804.02767>