A YOLOv8-Based System Object Detection and Localization

Ramazan YILDIZ
Department of Computer Engineering, Faculty of Engineering, Ataturk University, Erzurum, Turkey
Email: ry.yildizramazan@gmail.com
Telephone: +90 544 448 3169

Beyza GÜLER
Department of Computer Engineering, Faculty of Engineering, Ataturk University Erzurum, Turkey
Email: beyzafeyza.2001@gmail.com
Telephone: +90 552 647 5822

Abdelrahman MOHAMED
Department of Computer Engineering, Faculty of Engineering, Ataturk University, Erzurum, Turkey
Email: abdoessammo@gmail.com
Telephone: +90 552 750 8202

mdy
October 23, 2025

***Abstract*—This project addresses the fundamental computer vision challenge of accurate Object Detection and Localization across diverse categories, including humans, vehicles, animals, fruits, and inanimate objects. We utilize the modern YOLOv8 (You Only Look Once version 8) architecture, starting from a model pre-trained on the comprehensive COCO dataset, which recognizes approximately 80 object classes. To enhance the model's capability, we will extend its training by introducing 20–30 additional custom object classes, enabling it to detect a broader range of items. The OpenCV library is integrated for practical real-time video handling, preprocessing, and the crucial step of visualizing the predicted bounding boxes. Additionally, a Streamlit-based web application is developed to provide an interactive platform where users can perform real-time detection and view results directly through a browser. The core objective is to develop a working prototype that successfully integrates YOLOv8, OpenCV, and Streamlit, while also gaining hands-on experience in model fine-tuning, evaluation, deployment, and web-based implementation of deep learning systems with a focus on achieving a reasonable mAP (mean Average Precision).**

***Index Terms*—YOLOv8, Object Detection, Localization, Computer Vision, OpenCV, mAP, COCO Dataset, Streamlit, Real-time Detection.**

I. Dataset

This project uses the COCO (Common Objects in Context) dataset, a large-scale collection containing over 200,000 labeled images and 80 object categories such as humans, animals, and vehicles. The YOLOv8 model used in this study is initially pre-trained on the COCO dataset, providing a strong foundation for accurate and generalized object detection. To further improve and expand its recognition capability, we will fine-tune the model by adding 20–30 new custom object classes beyond the original COCO categories. For testing and evaluation, both the original COCO-based objects and the newly introduced custom objects will be analyzed in a real-time detection environment using video streams captured from cameras or public sources, ensuring robust assessment across diverse object types and real-world conditions.

II. Problem Statement

A. Problem to Be Solved

This project develops a real-time object detection prototype using YOLOv8 and OpenCV, supporting 80 COCO dataset categories including people, vehicles, animals, and everyday objects. The architecture is designed to be extensible through transfer learning for specialized applications.

B. Importance of the Problem

Fast and accurate object detection is critical in security, autonomous driving, industrial automation, agriculture, and smart cities. The system's scalability enables adaptation to specialized industry needs, offering both immediate practical utility and long-term adaptability across diverse domains.

III. Proposed Methodology

The project follows these main steps:

1) Utilizing the YOLOv8 architecture pre-trained on the COCO dataset (80 categories) and fine-tuning it with 20–30 additional custom object classes to expand detection capability.
2) Capturing and preprocessing real-time video frames using OpenCV to ensure compatibility with the model's input format.
3) Applying the fine-tuned YOLOv8 model to detect and localize multiple objects within the real-time video stream.
4) Visualizing detection results by drawing bounding boxes and class labels on the live video feed.
5) Developing a Streamlit-based web interface for real-time detection with browser-based result visualization.
6) Evaluating performance based on detection accuracy, bounding box correctness, and improvements in mean Average Precision (mAP) after fine-tuning.

Key Roles and Responsibilities of Team Members

Team Member	Key Responsibilities
Ramazan Yıldız	• Computer Vision Core Tasks (Model Configuration/Initialization and Training). • Web Application Development (Web Development with Streamlit, Image Management). • Documentation. • Version Control (Git, GitHub). • Debugging and Optimization (Computer Vision, Web Backend).
Beyza Güler	• Computer Vision Core Tasks (Model Training and Processing of Detection Results). • Testing and Evaluation (mAP calculation, performance analysis). • Documentation. • Version Control (Git, GitHub). • Debugging and Optimization (Focus on Computer Vision logic).
Abdelrahman Mohamed	• Computer Vision Core Tasks (Model Training OpenCV Image Preprocessing and Visualization). • Web Application Development (User Interface Design, Image Upload Functionality). • Documentation. • Version Control (Git, GitHub). • Debugging and Optimization (Computer Vision, Web Frontend/Display).

IV. Evaluation

The performance of the object detection system will be evaluated using several key metrics. These include Precision, Recall, mean Average Precision (mAP), and Intersection over Union (IoU). Precision and Recall will assess the model's ability to correctly identify and classify objects, while mAP will provide an overall measure of detection accuracy across all categories. IoU will be used to evaluate how accurately the predicted bounding boxes align with the actual object locations. Additionally, the average processing time per frame may be measured to assess the system's efficiency in the real-time environment.

V. Timeline

The project will be carried out over the next few weeks. First, the objectives, tools, and development environment will be prepared. Next, real-time video data will be collected and preprocessed using OpenCV for both training and testing purposes. The pre-trained YOLOv8 model will then be integrated and fine-tuned by adding 20–30 new custom object classes to extend its detection capability. After training, the fine-tuned model will be used to perform real-time object detection on live video streams, and the results will be visualized. A Streamlit-based web interface will be developed to allow users to perform real-time detection and view results directly through the browser. Finally, the system will be evaluated using mAP, Precision, Recall, and IoU metrics to compare the performance of the pre-trained and fine-tuned models, and the project will conclude with debugging, optimization, report writing, and preparation of the final presentation.

VI. Backup Plan

In case of unexpected issues during the project, the following backup measures will be implemented. If YOLOv8 or OpenCV encounters technical problems, a previous stable version of the model or library will be used, or alternative frameworks such as YOLOv5 may be considered. If difficulties arise during the fine-tuning process, such as insufficient data quality, training instability, or convergence issues, the backup plan includes retraining the model with a reduced number of custom object classes or a smaller subset of the dataset to ensure a functional version of the model can still be produced. Alternatively, hyperparameters such as learning rate, batch size, or number of epochs will be adjusted to stabilize the training. If some video data are corrupted or unavailable, additional publicly available video datasets or newly collected samples will be used to guarantee sufficient training and testing material. If the current system cannot handle computational load efficiently, the training and inference processes will be migrated to cloud-based platforms or higher-specification machines. If Streamlit does not work properly with the model, alternative web application development libraries will be considered to ensure a stable and functional user interface. In case of unexpected delays, the timeline will be adjusted by overlapping less critical steps or extending the schedule while maintaining the main project objectives as the priority.

Appendix

The full source code of the project is available at GitHub repository.