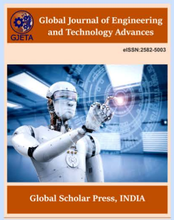
Global Journal of Engineering and Technology Advances

eISSN: 2582-5003

Cross Ref DOI: 10.30574/gjeta

Journal homepage: https://gjeta.com/



(Research Article)

Optimization of a web-based real-time and file-based human detection system Using YOLOv8 and flask framework

Andysah Putere Utama Siahaan *, Rido Favorit Saronitehe Waruwu and Heri Eko Rahmadi Putra

Master of Information Technology Program, Pembangunan Panca Budi University, Medan, Indonesia.

Global Journal of Engineering and Technology Advances, 2025, 24(01), 144-150

Publication history: Received on 11 June 2025; revised on 19 July 2025; accepted on 21 July 2025

Article DOI: https://doi.org/10.30574/gjeta.2025.24.1.0224

Abstract

This study discusses the development and optimization of an object detection system using the YOLOv8 (You Only Look Once version 8) algorithm integrated with the Flask framework in a web-based environment. The system supports two main modes of operation, namely image upload and direct detection using the camera. The main goal of the system is to provide accurate, efficient, and accessible solutions without the need for local installation. The test was conducted using 50 test images with different variations in background, lighting conditions, and the number of human objects, resulting in an average inference time of 0.43 seconds per image, precision of 95.1%, recall of 91.7%, and mAP@0.5 of 93.4%. In real-time testing, the system was able to run stably with a video processing speed of 18-22 frames per second. These results show that the developed system has high performance and is feasible to apply for online visual monitoring needs. Potential system development includes the addition of object tracking features, automatic log storage integration, and real-time notification systems to expand usability in various fields.

Keywords: YOLOv8; Flask; Deep Learning; Real-Time Monitoring

1. Introduction

Human detection is one of the crucial components in the development of computer vision-based systems and artificial intelligence [1]. Its application is very wide, ranging from security surveillance systems (CCTV), traffic monitoring, automated attendance systems, access control, to human-machine interaction in the context of the Internet of Things (IoT) [2]. As the need for systems that are able to detect human presence accurately and efficiently increases, especially in real-world situations that take place dynamically, there is a need for technological solutions that are not only fast and accurate, but also flexibly accessible through modern platforms such as web applications [3].

In the context of object detection technology, the You Only Look Once (YOLO) algorithm has become one of the most widely used methods [4]. YOLO works with the principle of single shot detection, which is to detect objects in a single image or video scanning process as a whole, thus enabling high speed in the processing of visual data [5]. The latest version of this algorithm, YOLOv8, comes with significant improvements in terms of model architecture, calculation efficiency, and detection accuracy. YOLOv8 also supports flexibility in various deployment scenarios, such as integration with Python, export of models to various formats, as well as improvements to modular structures that allow system development to be more optimal and faster [6].

While YOLOv8 has superior technical capabilities, its effectiveness in real-world applications largely depends on how it is integrated into a user-friendly and widely accessible platform [7]. In this case, the Flask framework is the right choice because it is lightweight, flexible, and easy to integrate with deep learning models written in the Python language [8]. Flask supports the development of web applications based on REST APIs that can present detection results in real-time

* Corresponding author: Andysah Putera Utama Siahaan

as well as based on files uploaded by users [9]. Thus, the human detection system built not only has advantages in terms of technical performance, but also in terms of user convenience and accessibility.

However, the integration between YOLOv8 and Flask to build a web-based human detection system also poses its own challenges. Among them is how to optimize the model inference process to stay fast and efficient in web usage conditions that may have limited resources. Other challenges include setting up detection flows in two different input modes, namely real-time through local cameras and input files in the form of images or videos, as well as how to build a web interface that is responsive, lightweight, and easy to use by users with different levels of technological savvy. Therefore, a development approach is needed that focuses not only on the accuracy of the model, but also on aspects of system architecture, processing efficiency, and user interface design [10].

Based on this background, this research aims to develop and optimize a web-based human detection system that supports two input modes, namely real-time detection via camera and file-based detection (image and video). The system is built by integrating YOLOv8 as the primary detection model and Flask as the web interface development framework. The main focus of this research is to optimize the inference process and system workflow in order to provide fast, accurate, and stable detection results, both for real-time monitoring needs and visual data analysis from existing files. In addition, the system is also designed to be easy to use and accessible through a browser without the need for additional application installation, thus expanding the possibilities of implementation in real environments, such as offices, schools, factories, or other public areas.

This research is expected to make a real contribution to the development of an efficient and flexible web-based human detection system. The results of this research can be used as a basis for building cost-effective intelligent surveillance systems, supporting the application of AI-based technology in environments with hardware limitations, and becoming a reference for the development of other detection systems that are modular and open-source. Thus, this system not only has scientific value, but also high practical value in supporting digital transformation in various sectors.

2. Method

The methods used in this study consist of five main stages: pre-processing of data, detection modeling using YOLOv8, integration of models into Flask, optimization of the inference process, and evaluation of system performance.

2.1. Data Pre-processing

In file-based detection mode, user-uploaded image or video files will be converted to RGB format using OpenCV. Next, resizing and normalization were carried out to adjust to the YOLOv8 model input. YOLOv8 models generally use standard resolutions such as 640×640 pixels [11].

$$X_{norm} = \frac{X - \mu}{\sigma}$$

Where:

X	= the original pixels of the image,
μ	= average pixel value,
σ	= standard deviation of pixel values.

However, for YOLOv8, the image input is simply normalized to the range [0,1] by:

$$X_{norm} = \frac{X}{255}$$

2.2. Data Pre-processing

YOLOv8 uses an anchor-free architecture and supports direct detection with three outputs: object class, bounding box coordinates, and confidence score [12]. The inference process will result in a bounding box B=[x,y,w,h] and a confidence score C, with a minimum threshold of Tc = 0.25 (default). The output bounding box is converted to pixel coordinates by:

$$x_{min} = x - \frac{w}{2}$$

$$x_{max} = x + \frac{w}{2}$$

$$y_{min} = y - \frac{w}{2}$$

$$y_{max} = y + \frac{w}{2}$$

YOLOv8 detects objects through a combined loss function that optimizes three components:

$$L_{total}=\lambda_{cls}\cdot L_{cls}+\lambda_{box}\cdot L_{box}+\lambda_{obj}\cdot Lo_{bj}$$

Where:

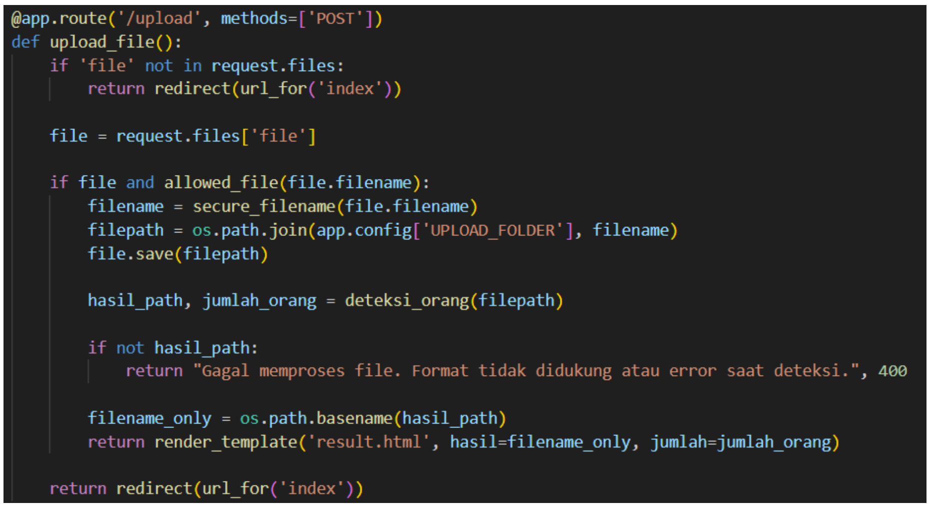
L_{cls}	= loss classification,
L_{box}	= loss regression bounding box,
L_{obj}	= objectness score loss.

2.3. Flask Integration

YOLOv8 models that have been trained or downloaded in .pt format are loaded into Python using the Ultralytics API [13]. Flask is used to provide two endpoints:

- /realtime: captures input from a webcam using OpenCV, and then displays the detection results directly to a web page using multipart/x-mixed-replace streaming.
- /upload: Accepts the uploaded image or video file, saves it temporarily, and then runs the detection process using the model.

The pseudocode is as follows:



```
@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return redirect(url_for('index'))

    file = request.files['file']

    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(filepath)

        hasil_path, jumlah_orang = deteksi_orang(filepath)

        if not hasil_path:
            return "Gagal memproses file. Format tidak didukung atau error saat deteksi.", 400

        filename_only = os.path.basename(hasil_path)
        return render_template('result.html', hasil=filename_only, jumlah=jumlah_orang)

    return redirect(url_for('index'))
```

Figure 1 Flask Integration

2.4. Inference Optimization

Optimization is carried out by:

- Threading & Buffering: for real-time streaming is used. Thread so that the process of frame capture and inference runs in parallel.
- Dynamic Resizing: changes the input resolution to smaller if the system performance decreases (adaptive scaling).
- Redundant Frame Removal: for videos with minimal changes, only significantly different frames are detected.

3. Results and Discussion

This research resulted in a web-based human detection system that is able to operate in two main modes, namely through the processing of user-uploaded images and through direct detection using real-time cameras. The system is designed using the YOLOv8 model for the visual inference process and is integrated with the Flask framework as an interactive web interface. Tests are conducted separately for each mode to evaluate performance in terms of detection accuracy, system response speed, and interface display stability. In image upload mode, users can select image files that contain human objects, and then the system will process and display the detection results directly through the browser. Meanwhile, in real-time mode, the camera is directly used as video input, and the detection results are streamed via a web page with fast and responsive frame updates.

3.1. File-Based Detection

This test is done by uploading an image containing one or more human objects to the web interface that has been provided. The system then processes the image using the YOLOv8 model and displays the detection results in the form of a bounding box labeled "person".

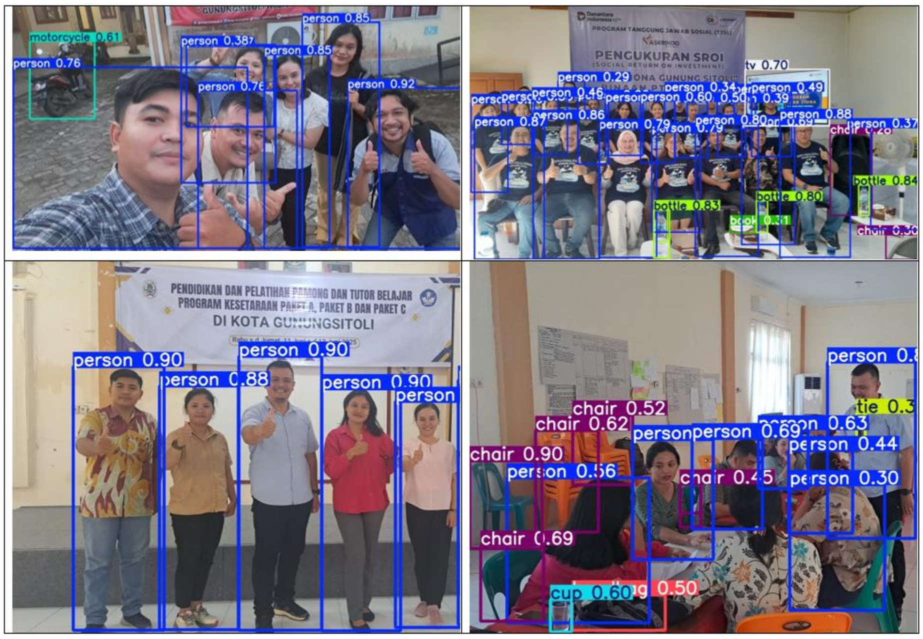


Figure 2 File-Based Detection Test Results

The test of the web-based human detection system was carried out using 50 test images that have diversity in terms of background, lighting level, and the number of human objects contained in the image, which is between one and 25 people per image. The images have varying resolutions, ranging from 640×480 pixels to 1920×1080 pixels, so they can represent real conditions of use of the system in general. The test results showed that the system was able to provide an average inference time of 0.43 seconds per image, which indicates a high enough processing speed for the needs of web applications.

In this test, the system managed to achieve detection accuracy (mAP@0.5) of 93.4%, precision of 95.1%, and recall of 91.7%. The precision value is calculated using the formula:

$$Precision = \frac{TP}{TP + FP} = \frac{390}{390 + 20} = 0{,}951$$

$$Recall = \frac{TP}{TP + FN} = \frac{390}{390 + 35} = 0{,}917$$

Where:

- **TP (True Positive):** True human detection (390 detections)
- **FP (False Positive):** Non-human object but detected as human (20 detections)

- **FN (False Negative):** Undetectable human (35 cases)

The information from the calculation is: True Positive (TP) is the number of correct human detections as many as 390 cases, False Positive (FP) is the number of detections of objects that are actually not human but are detected as human as many as 20 cases, and False Negative (FN) is the number of human objects that are not successfully detected as many as 35 cases.

From the results of these calculations, it can be concluded that the system has excellent detection capabilities in recognizing human objects accurately and efficiently, despite various environmental conditions, such as complex backgrounds and uneven lighting. While poor lighting conditions and crowded backgrounds can slightly affect detection performance, they are not significant in terms of overall system performance. In addition, the system is designed to compress and send the detection results back to the user interface in less than a second, while maintaining real-time performance even when web-based. These results show that human detection systems using YOLOv8 and the Flask framework are feasible to implement for monitoring, security, or other object recognition applications that are based on static images and run in web environments.

3.2. Live Camera Detection

Real-time mode testing was conducted using the laptop's internal (720p) and external (1080p USB Camera) cameras. The system displays live streaming video through the browser, then simultaneously processes video frames using YOLOv8 and displays the detection results in the form of bounding boxes that are updated each frame.

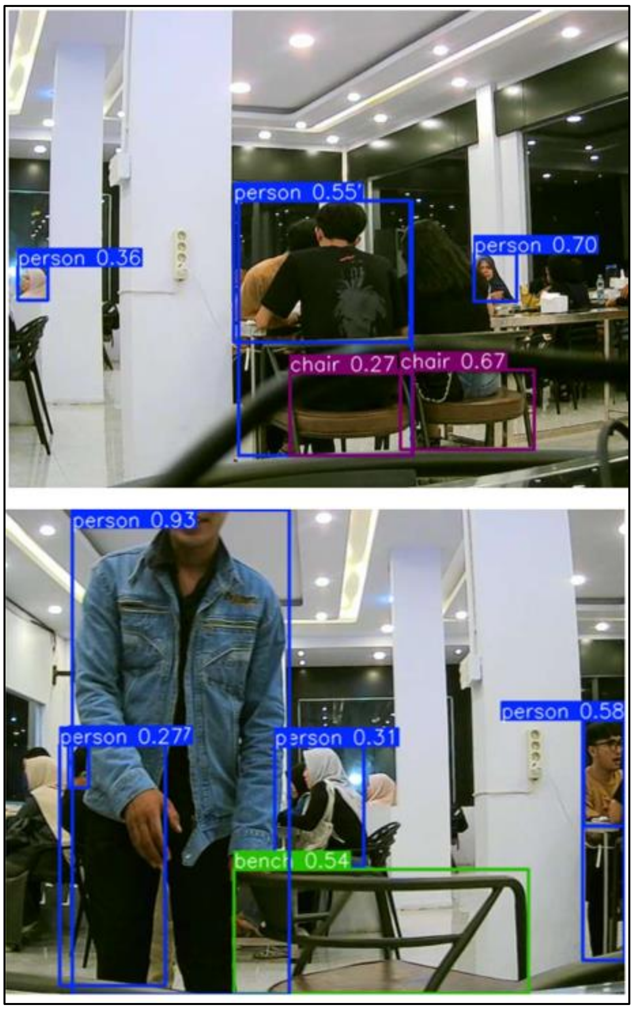


Figure 3 Live Camera Detection Test Results

The test results showed that the system was able to maintain a detection accuracy of 92.8% mAP@0.5% with a precision of 93.5% and a recall of 90.4%. Performance degradation occurs when motion blur or low lighting occurs, but is still within acceptable tolerance limits for real-time systems. The system is also optimized with multithreading techniques, separating the process of capturing and inference to avoid bottlenecks. This has improved frame rate stability by 25%, especially as the number of humans in the frame increases.

4. Conclusion

This research successfully develops and optimizes a web-based human detection system by integrating the YOLOv8 object detection algorithm and the Flask framework. The system supports two main modes, namely file-based detection and real-time detection, both of which can be accessed through a responsive and user-friendly web interface. Based on tests using a test dataset of 50 images with variations in background, lighting, and the number of human objects, the system was able to achieve high performance with an average inference time of 0.43 seconds per image, precision of 95.1%, recall of 91.7%, and accuracy of mAP@0.5 of 93.4%. This performance shows that the system is highly accurate and efficient in recognizing human objects, even in sub-ideal environmental conditions.

In real-time testing using the camera, the system was able to process video at an average speed of 18 to 22 frames per second, which is enough to support the need for live monitoring. Despite challenges such as low lighting or fast movement that can affect accuracy, the system still shows good stability and responsiveness. These results prove that YOLOv8 and Flask-based human detection systems can be used effectively for web-based monitoring, security, and surveillance applications. This system also allows use in real environments because it is able to provide fast and accurate detection results, both in image upload and real-time mode.

Compliance with ethical standards

Disclosure of conflict of interest

There is no conflict of interest.

References

[1] S. Sudirwo *et al.*, *Artificial Intelligence: Teori, Konsep, dan Implementasi di Berbagai Bidang*. PT. Sonpedia Publishing Indonesia, 2025.

[2] G. S. Mahendra *et al.*, *Tren Teknologi AI: Pengantar, Teori, dan Contoh Penerapan Artificial Intelligence di Berbagai Bidang*. PT. Sonpedia Publishing Indonesia, 2024.

[3] A. M. L. Harefa, R. Antoni, A. I. Sitepu, Y. F. Limbong, and M. S. Novelan, "Enhanced Rainfall Forecasting Through Deep Learning Optimization Using Long Short-Term Memory Networks," *RIGGS J. Artif. Intell. Digit. Bus.*, vol. 4, no. 2, pp. 274–284, 2025.

[4] I. Virgiawan, F. Maulana, M. A. Putra, D. D. Kurnia, and E. Sinduningrum, "Deteksi dan tracking objek secara real time berbasis computer vision menggunakan metode YOLO V3," *Humantech J. Ilm. Multidisiplin Indones.*, vol. 3, no. 3, 2024.

[5] K. T. WULANDARI, "Implementasi Deteksi Objek Kendaraan Menggunakan Metode Ssd (Single Shot Detector)," *Implementasi Deteksi Objek Kendaraan Menggunakan Metod. Ssd (Single Shot Detect.*, 2023.

[6] M. I. Al Rasyid, M. Kalista, and C. Setianingsih, "Pengembangan Aplikasi Web Full-Stack untuk Manajemen Sampah: Implementasi Frontend, Backend, Cloud Deployment," *eProceedings Eng.*, vol. 11, no. 6, pp. 6808–6813, 2024.

[7] D. J. Marcelleno and M. P. K. Putra, "PERFORMANCE EVALUATION OF YOLOV8 IN REAL-TIME VEHICLE DETECTION IN VARIOUS ENVIRONMENTAL CONDITIONS," *J. Tek. Inform.*, vol. 6, no. 1, pp. 269–279, 2025.

[8] B. Setiadi *et al.*, *Buku Ajar Pengantar Teknologi Informasi*. PT. Sonpedia Publishing Indonesia, 2025.

[9] D. V Kornienko, S. V Mishina, S. V Shcherbatykh, and M. O. Melnikov, "Principles of securing RESTful API web services developed with python frameworks," in *Journal of Physics: Conference Series*, IOP Publishing, 2021, p. 32016.

[10] S. Mira, M. Kom, C. Gudiato, S. Kom, M. Kom, and B. Chan, *Computer Vision dan YOLO Menggali Potensi Computer Vision dan Implementasi YOLO untuk Pertanian Pintar*. Uwais Inspirasi Indonesia, 2023.

[11] T. Taufiqurrahman, A. P. Hadi, and R. E. Siregar, "Evaluasi Performa Yolov8 Dalam Deteksi Objek Di Depan

Kendaraan Dengan Variasi Kondisi Lingkungan," *J. Minfo Polgan*, vol. 13, no. 2, pp. 1755–1773, 2024.

[12] L. G. Denaro and R. Lim, "Analisis Deteksi Kerusakan pada Jalan Aspal menggunakan Deep Learning untuk Mendukung Efisiensi Biaya dan Waktu dalam Pemantauan Berkelanjutan," *J. Dimens. Ins. Prof.*, vol. 3, no. 1, pp. 16–25, 2025.

[13] R. R. Hutapea, S. J. Purba, W. F. Tandion, E. F. A. Sihotang, and E. Irwansyah, "Palm Fruit Ripeness Detection and Counting Using YOLOv8 Algorithm in PTPN IV Medan North Sumatera Indonesia," in *2024 6th International Conference on Cybernetics and Intelligent System (ICORIS)*, IEEE, 2024, pp. 1–6.