Real-Time Object Detection Using YOLO

Kanishkaa K,
B.Tech Artificial Intelligence and Data Science,
Sri Shakthi Institute of Engineering and Technology,
Coimbatore.
kanishkaakrishnakumar@gmail.com

Mithun R,
B.Tech Artificial Intelligence and Data Science,
Sri Shakthi Institute of Engineering and Technology,
Coimbatore.
mithunrajan60@gmail.com

Ramani R,
B.Tech Artificial Intelligence and Data Science,
Sri Shakthi Institute of Engineering and Technology,
Coimbatore.
ramani13177@gmail.com

Santhoshi P,
Assistant Professor,
Department of Artificial Intelligence and Data Science,
Sri Shakthi Institute of Engineering and Technology,
Coimbatore.
Santhoshiaids@siet.ac.in

ABSTRACT:

Real-time object detection is a crucial task in computer vision that enables the identification and localization of objects in images and video frames. It is widely used in applications such as autonomous driving, surveillance, traffic monitoring, and robotics, where real-time processing is essential. The You Only Look Once (YOLO) model is a state-of-the-art deep learning approach known for its speed and accuracy. Unlike traditional region-based methods, YOLO processes an entire image in a single pass using a fully convolutional network, dividing it into a grid and predicting bounding boxes and class probabilities simultaneously, making it highly efficient for real-world applications. YOLOv7, the latest iteration, offers improved accuracy, faster inference, and better model efficiency. In this paper, we propose a customized YOLOv7-based real-time object detection model with various enhancements aimed at improving detection precision, computational efficiency, and adaptability for public utility. Our approach integrates advanced pre-processing, optimized training strategies, and fine-tuned hyper-parameters to enhance object recognition while ensuring real-time performance. Additionally, we incorporate techniques such as model quantization and pruning to optimize speed and reduce computational costs. The proposed model is well-suited for applications requiring high-speed and accurate detection, contributing to smart cities, surveillance, and traffic monitoring, making it a valuable asset in modern technological advancements.

Keywords: Real-time object detection, YOLOv7, deep learning, computer vision, bounding box prediction, surveillance, traffic monitoring, smart cities, real-time processing.

I. INTRODUCTION

Real-time object detection plays a crucial role in modern computer vision, enabling the identification and localization of objects in images and video streams. It is widely applied in areas such as autonomous vehicles, surveillance, traffic monitoring, robotics, industrial automation, and smart city solutions. Traditional region-based object detection methods, such as Region-based Convolutional Neural Networks (R-CNN) and its variants, rely on multi-stage processes that include region proposal generation, feature extraction, and classification. While effective, these methods often suffer from high computational costs and slower inference speeds, making them impractical for real-time applications. The emergence of single-stage detectors, particularly the You Only Look Once (YOLO) model, has significantly

transformed the landscape of real-time object detection by addressing these challenges. Unlike traditional approaches, YOLO processes an entire image in a single pass using a fully convolutional network, dividing it into grids and predicting bounding boxes and class probabilities simultaneously. This single-shot detection mechanism drastically improves processing speed and efficiency, making YOLO a widely adopted choice for real-world applications requiring high-speed object recognition.

The latest iteration, YOLOv7, introduces multiple improvements over its predecessors, including enhanced detection accuracy, better feature extraction, optimized inference speed, and improved model efficiency. YOLOv7 incorporates architectural advancements such as extended efficient layer aggregation networks (E-ELAN) and dynamic label assignment, allowing it to achieve superior performance in object detection tasks. Furthermore, the model employs more effective computational optimization strategies, making it capable of running efficiently even on resource-constrained edge devices. These improvements make YOLOv7 highly suitable for real-time applications, where both speed and accuracy are critical. However, while YOLOv7 demonstrates outstanding performance, further optimizations can be implemented to enhance its adaptability for specific use cases. In this paper, we propose a customized YOLOv7-based real-time object detection model that incorporates multiple refinements, including advanced data pre-processing, optimized training methodologies, and fine-tuned hyper-parameters. These modifications are designed to maximize detection accuracy while ensuring minimal computational overhead, making the model more suitable for deployment in real-world environments.

The proposed model is tailored for intelligent surveillance, industrial automation, public safety, and traffic monitoring systems, where rapid and precise object detection is crucial. By integrating additional techniques such as model quantization and pruning, we further optimize the speed and efficiency of our approach, reducing the computational footprint without sacrificing accuracy. Our model also incorporates domain-specific enhancements that improve object recognition in challenging conditions, such as low-light environments, occlusion, and motion blur. The objective of this research is to develop an efficient, robust, and adaptable real-time object detection framework that can be seamlessly integrated into various smart applications. The rest of this paper is structured as follows: Section II discusses related work and background, Section III & IV details our proposed solution and methodology, Section V presents experimental results, and Section VI concludes with insights and future research directions.

II. LITERATURE REVIEW

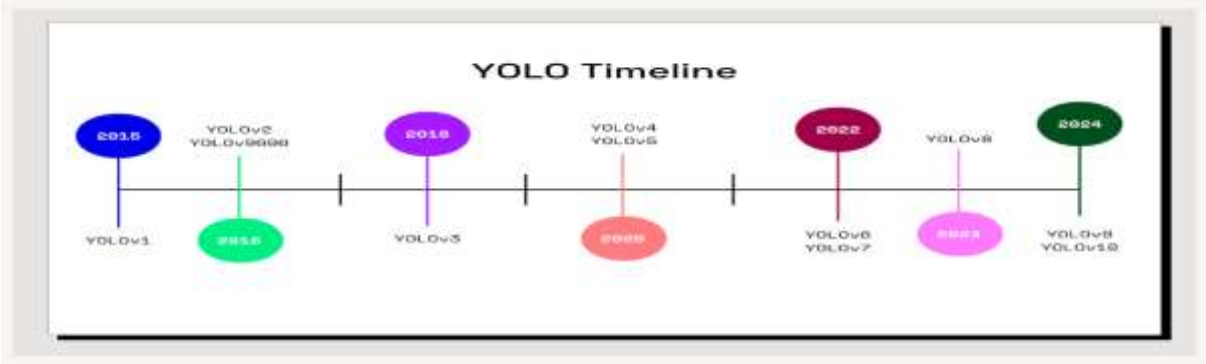
A comprehensive literature review would delve into these aspects in greater detail and cite specific research papers, studies, and articles.

1. Evolution of Object Detection

Early object detection models like R-CNN (Girshick et al., 2014), Fast R-CNN (Girshick, 2015), and Faster R-CNN (Ren et al., 2015) relied on region proposal networks, making them computationally expensive. These models performed well in accuracy but lacked efficiency for real-time applications. The introduction of YOLO (Redmon et al., 2016) revolutionized detection by using a single-stage approach, significantly reducing inference time.

2. YOLO Architecture and Advancements

YOLO evolved from YOLOv1 to YOLOv7, incorporating grid-based prediction, anchor-free detection, and optimized convolutional layers, enhancing both accuracy and inference speed.



3. Comparison with Other Object Detection Models

Object detection has progressed with models like Faster R-CNN, SSD, and YOLO. Faster R-CNN (Ren et al., 2015) delivers high accuracy but is computationally demanding, while SSD (Liu et al., 2016) improves speed but sacrifices some precision. YOLO (Redmon et al., 2016) balances speed and accuracy, making it suitable for real-time applications. The latest version, YOLOv7 (Wang et al., 2022), introduces dynamic label assignment and model compression for enhanced efficiency.

4. Applications of Real-Time Object Detection

Real-time object detection is widely applied in autonomous vehicles for pedestrian and obstacle recognition (Geiger et al., 2013), smart surveillance for anomaly detection (Bochkovskiy et al., 2020), and traffic monitoring in smart cities (Jocher et al., 2023). In healthcare, deep learning aids in medical imaging analysis (Litjens et al., 2017), while industrial automation benefits from robotic vision for defect detection.

5. Challenges in Real-Time Object Detection

Despite advancements, challenges such as occlusion, small object detection, and lighting variations impact detection accuracy (Zhu et al., 2020). Solutions involve feature fusion, attention mechanisms, and adaptive learning techniques.

6. Enhancements in YOLOv7 for Practical Use

YOLOv7 introduces dynamic label assignment, model compression techniques, and efficient training strategies, improving its adaptability for real-time applications.

7. Role of Deep Learning in Object Detection

Convolutional Neural Networks (CNNs) form the backbone of modern object detection models. Deep learning enables hierarchical feature extraction, improving detection precision across different object scales. The synergy between deep learning and YOLO architecture continues to drive advancements in object detection.

8. Optimization Techniques for YOLOv7

Methods like quantization, pruning, and knowledge distillation (Han et al., 2015) help reduce computational costs while maintaining accuracy, making YOLOv7 viable for edge AI deployment.

9. Future Trends in Object Detection

The field is shifting towards self-supervised learning, edge AI, and hybrid transformer-based models (Dosovitskiy et al., 2020) for improved accuracy, lower latency, and energy-efficient.

III. PROPOSED SOLUTION

Our YOLO detection model incorporates several advancements and customizations to enhance usability for the public. We utilize YOLOv7, the state-of-the-art real-time object detector, which surpasses all known object detectors in both speed and accuracy. For improved clarity, object detection is displayed in full screen with a user-friendly GUI. To enhance comprehension, we use a consistent colour scheme for bounding boxes per class (e.g., blue for buses, yellow for persons). Additionally, the model displays the count of detected objects (e.g., bus = 2, persons = 5) on the top left of the screen, which is also printed in the console when the 'ESC' key is pressed—useful for applications like traffic monitoring and surveillance. The FPS (Frames Per Second) is displayed on the top right to provide performance insights. Our model supports object detection from video and image files (--source file_name.mp4/jpg) as well as real-time detection via webcam (--source 0), making it versatile for various use cases.

IV. METHODOLOGY

1. **Command-Line Argument Parsing**

The implementation uses Python's argparse module to allow users to specify model parameters and input sources dynamically. This enables flexibility in configuring the model based on different use cases. Users can pass arguments such as the model's weight file, image size, confidence threshold, and device type (CPU/GPU).

Example usage:

```
PS C:\Users\KAVISHKAA\Documents\customyolov7> python test1.py --weights yolov7.pt --source video1.mp4 --img-size 640 --conf-thres 0.25
>>
```

This allows for seamless customization of detection parameters without modifying the script manually.

2. **Device Selection and Model Initialization**

The script detects whether a GPU is available and automatically selects the appropriate device (CUDA for GPU or CPU otherwise). The YOLOv7 model is then loaded using the attempt_load() function, which initializes the network with pre-trained weights. If a GPU is detected, the model operates in FP16 precision, which significantly improves computational efficiency and reduces memory usage. If the --no-trace argument is not set, a traced model is created for optimization, ensuring smooth inference.

3. **Input Data Handling**

The implementation supports multiple input sources:

- **Webcam:** Uses LoadStreams to continuously fetch frames from the camera in real-time.
- **Video/Image Files:** Uses LoadImages to read frames sequentially from a given file path.

Each input frame is resized to match the model's expected dimensions using check_img_size(). This ensures compatibility with YOLOv7's architecture while preserving aspect ratios as much as possible.

4. **Data Preprocessing and Normalization**

Before inference, input images undergo preprocessing:

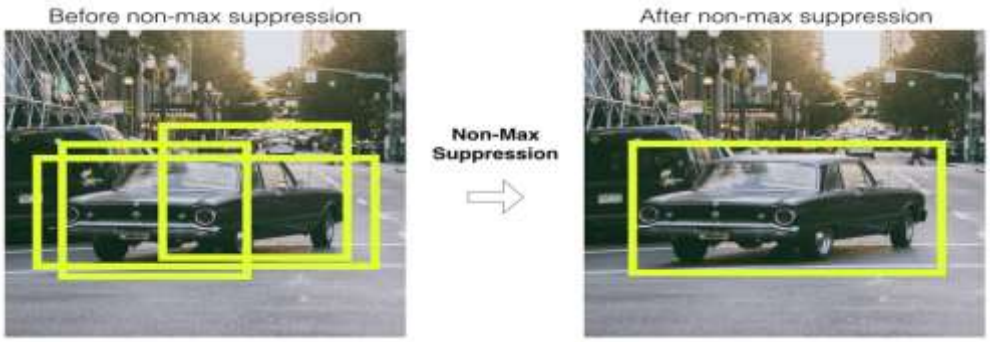
- Conversion to a PyTorch tensor using torch.from_numpy().
- Precision adjustment (FP16 if using GPU, FP32 otherwise).
- Normalization by scaling pixel values to the range [0, 1].

These steps ensure consistency and improve detection accuracy while optimizing computational performance.

5. **Object Detection and Non-Maximum Suppression (NMS)**

Once an image is pre-processed, it is passed through the YOLOv7 model for inference. The raw output contains multiple overlapping bounding boxes for detected objects. Non-maximum Suppression (NMS) is applied to remove redundant boxes, ensuring only the most relevant detections remain. This is controlled by two key parameters:

- **Confidence Threshold (--conf-thres)**: Filters out weak detections.
- **IOU Threshold (--iou-thres)**: Determines the overlap threshold for merging similar detections.



6. **Bounding Box Scaling and Object Classification**

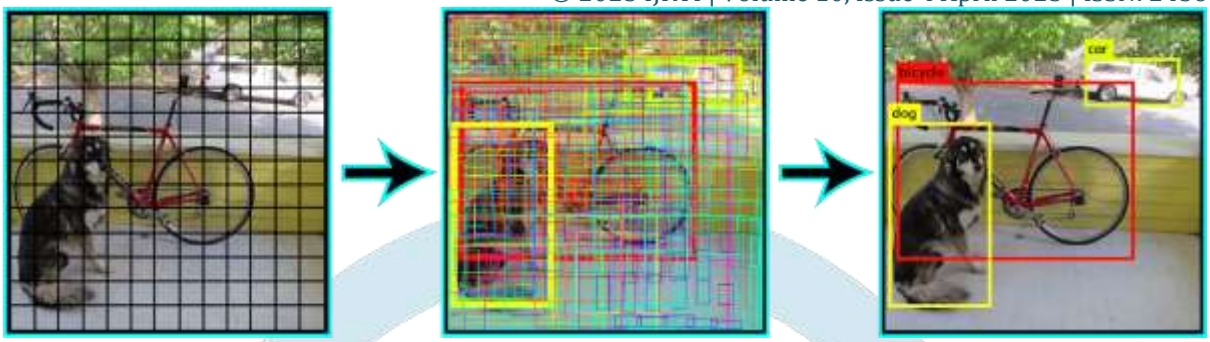
After applying NMS, detected bounding boxes are scaled back to the original image dimensions using scale_coords(). Each detection is then classified into one of the pre-defined classes, with class names obtained from the YOLO model's metadata. A dictionary is maintained to store the number of detections per class.

7. **Class-Wise Object Counting**

The script keeps track of detected objects per class using euclidean distance and updates a cumulative count.This count is displayed on the screen in real-time using OpenCV's cv2.putText().The final object count is printed to the console when the user exits the program.

8. **Drawing Bounding Boxes and Labels**

Detected objects are highlighted with bounding boxes, and the corresponding class label along with confidence score is overlaid on each box. This is handled by the plot_one_box() function, which ensures clear visibility of detected objects. Different colors are used for each class to enhance distinction. Each detected object class is assigned a unique color for easy differentiation.



9. **FPS Calculation for Performance Monitoring**

The system dynamically calculates FPS using OpenCV's cv2.getTickCount() and cv2.getTickFrequency(), which measure the time elapsed between consecutive frames. The FPS value is displayed on the screen in the top-right corner, providing real-time feedback on processing speed.

```
curr_tick = cv2.getTickCount()
fps = cv2.getTickFrequency() / (curr_tick - prev_tick)
prev_tick = curr_tick
```

This metric helps assess the efficiency of the model and hardware configuration.

10. **Full-Screen Visualization using OpenCV**

For better user experience, the output is displayed in a full-screen OpenCV window using:

```
cv2.namedWindow("Real-Time Detection", cv2.WND_PROP_FULLSCREEN)
cv2.setWindowProperty("Real-Time Detection", cv2.WND_PROP_FULLSCREEN, cv2.WINDOW_FULLSCREEN)
```

This ensures that detected objects and information overlays are clearly visible without window constraints.

11. **Exit Handling and Final Object Count Summary**

The detection process continues until the user presses the 'Esc' key (cv2.waitKey(1) & 0xFF == 27). Upon exit, the final count of each detected object class is printed to the console, summarizing the detection session.

```
print("\nFinal Count of Detected Objects:")
for obj, count in total_count.items():
    print(f"{obj} = {count}")
```

This provides a comprehensive overview of the objects detected throughout the session.

V. RESULT


```
customyolov7\video1.mp4: video 1/1 (19/513) C:\Users\KANISHKAA\Documents\customyolov7\video1.mp4:
Final Count of Detected Objects:
person = 6
car = 5
bus = 2
truck = 2
traffic light = 1
motorcycle = 1
dog = 1
horse = 1
PS C:\Users\KANISHKAA\Documents\customyolov7>
```

VI. CONCLUSION AND FUTURE ENHANCEMENTS

Real-time object detection using YOLO (You Only Look Once) has revolutionized computer vision by enabling fast and accurate detection of multiple objects in a single pass through the neural network. Its efficiency stems from treating object detection as a single regression problem, directly predicting class labels and bounding boxes from input images. Compared to traditional detection models, YOLO is significantly faster, making it ideal for real-time applications such as autonomous driving, surveillance, and robotics. The integration of YOLO with edge devices and cloud computing expands its applicability in real-world scenarios. Overall, YOLO remains a

powerful and evolving tool in object detection, balancing speed and accuracy for diverse applications.

Our model has several advantages over classifier-based systems. Unlike region-based methods that perform classification on multiple proposed regions, YOLO processes the entire image at once, making its predictions more informed by the global context. Additionally, it operates with a single network evaluation, unlike R-CNN-based approaches that require thousands of evaluations per image. This results in an extremely fast detection pipeline, making YOLO over 1000 times faster than R-CNN and 100 times faster than Fast R-CNN. This efficiency allows the model to be deployed in real-time scenarios without the need for extensive computational resources, making it highly suitable for applications such as automated surveillance, pedestrian detection, and traffic management.

Potential future enhancements include:

- Fine-tuning the model with high-quality datasets and optimizing anchor boxes can enhance performance for specific tasks.
- Integration with self-supervised learning and domain adaptation techniques can enhance YOLO's performance in diverse environments without extensive retraining.
- Advancements like detecting bikes with more than two persons and detecting their license plate.

This implementation serves as a robust foundation for real-world applications such as surveillance, traffic monitoring, and retail analytics. The continuous evolution of YOLO models, coupled with on-going research in deep learning and optimization strategies, will further enhance the efficiency and accuracy of real-time object detection systems. Future developments in hardware acceleration, model compression, and adaptive learning techniques will play a crucial role in making object detection more accessible, scalable, and effective in various domains. By implementing the aforementioned enhancements, the proposed model can achieve even greater performance, solidifying its relevance in high-speed and high-precision applications. As the field of computer vision advances, the capabilities of real-time object detection will continue to grow, unlocking new possibilities in automation, security, and intelligent decision-making.

VII. REFERENCES

1. Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. (2022). Swin Transformer: Hierarchical vision transformer using shifted windows. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

2. Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. (2021). FAIRMOT: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision, 129(11), 3069–3087*.

3. Chien-Yao Wang, Alexey Bochkovskiy, Hong-Yuan Mark Liao. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*.

4. Mohammadamin Baghbanbashi, Mohsen Raji, Behnam Ghavami. (2024). Quantizing YOLOv7: A Comprehensive Study. *arXiv preprint arXiv:2401.12345*.

5. Xun Chen, Linyi Deng, Chao Hu, Tianyi Xie, Chengqi Wang. (2024). Dense small object detection based on an improved YOLOv7 model. *Neural Computing and Applications, 36(2), 1879–1892*.

6. Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. (2016). You Only Look Once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779–788*.

7. Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. (2020). YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

8. Glenn Jocher et al. (2023). Ultralytics YOLOv8: Cutting-edge real-time object detection and segmentation.
GitHub Repository,https://github.com/ultralytics/ultralytics.

9. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg. (2016). SSD: Single shot multibox detector. *European Conference on Computer Vision (ECCV), 21–37*.

10. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778*.

11. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 28, 91–99.

12. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. (2014). Microsoft COCO: Common objects in context. *European Conference on Computer Vision (ECCV)*, 740–755.