International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 08 | Aug – 2025 DOI: 10.55041/ISJEM04976
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Real-Time Object Detection Using Yolov5: A Transfer Learning Approach

[1]RONGALA RAJESH,[2]DEVANA SUREKHA

[1]Assistant Professor, Department Of MCA, 2MCA Final Semester,
[1]Master of Computer Applications,
[1]Sanketika Vidya Parishad Engineering College, Vishakhapatnam, Andhra Pradesh, India

Abstract:

Real-time object detection is essential in areas like surveillance, autonomous vehicles, and smart cities.This project uses YOLOv5, a fast and accurate one-stage deep learning model, for object detection.Transfer learning is applied by fine-tuning pre-trained weights on a custom dataset of 5,000 images.The model includes CSPDarknet53 and PANet to improve feature extraction and multi-scale detection.It effectively detects small, occluded, and multiple objects in complex scenes.Trained over 50 epochs, it achieved high accuracy using precision, recall, and mAP metrics.Deployment was done using OpenCV DNN for real-time CPU-based inference.The system shows strong real-world potential, with future scope in edge and IoT applications.

Index Terms: YOLOv5, Real-Time Object Detection, Deep Learning, Transfer Learning, CNN, CSPDarknet53, PANet, Pascal VOC, Bounding Box, Data Augmentation, OpenCV, Edge Deployment, Object Classification, Precision, Recall, mAP, F1-Score, Smart Surveillance, Autonomous Vehicles, Lightweight Model

1.Introduction:

Object detection is a key task in computer vision that involves identifying and locating objects in images or videos. It plays a vital role in applications such as autonomous driving, surveillance, and industrial automation. Traditional methods struggled with speed and accuracy, especially for real-time scenarios.

To address these challenges, YOLOv5, a deep learning-based one-stage detector, offers faster and more accurate object detection. It processes images in a single pass, making it ideal for real-time applications. This project leverages YOLOv5's architecture for efficient detection in diverse environments.

Transfer learning is used to fine-tune the model on a custom dataset, reducing training time and improving accuracy. Advanced components like CSPDarknet53 and PANet enhance detection of small and overlapping objects. The system is deployed using OpenCV for real-time inference on standard hardware.

1.1. Existing system

Earlier object detection systems relied on traditional techniques like Haar cascades, SVMs, and hand-crafted features such as HOG and SIFT. These methods were often slow, lacked robustness, and failed under complex conditions like occlusion or varied lighting. Deep learning models like R-CNN, Fast R-CNN, and Faster R-CNN improved accuracy but required high computational power and multi-stage processing.

One-stage detectors like SSD and the earlier versions of YOLO introduced faster inference but had limitations in detecting small or overlapping objects. These models also struggled with real-time performance on resource-constrained devices. Additionally, high false positives, poor generalization, and inefficiency in localization limited their use in critical real-world applications.

1.1.1.Challenges

Detection of Small and Occluded Objects

- Accurately identifying small, overlapping, or partially visible objects remains difficult.

Real-Time Processing on Low-Power Devices

- Achieving high-speed detection on edge devices like Raspberry Pi or Jetson Nano is challenging due to hardware limitations.

High False Positives and Localization Errors

- Misclassification and inaccurate bounding boxes can affect overall system performance.

Dataset Limitations

❖ Domain-specific datasets may lack diversity, leading to poor generalization in real-world scenarios.

Lighting and Environmental Variations

❖ Performance drops under low light, glare, shadows, or adverse weather conditions.

Model Size and Computational Cost

❖ Larger models provide better accuracy but are harder to deploy in real-time environments.

Dynamic Backgrounds and Motion Blur

❖ Object detection in videos with fast motion or complex backgrounds can reduce precision.

1.2 Proposed system:

The proposed system uses YOLOv5, a fast and accurate one-stage object detection model, to detect multiple objects in real-time.It incorporates CSPDarknet53 as the backbone for efficient feature extraction and PANet for multi-scale feature refinement.[8]These improvements help in detecting small, overlapping, and occluded objects in complex scenes.To enhance accuracy and reduce training time, transfer learning is applied using pre-trained YOLOv5 weights.A custom dataset of 5,000 annotated images is used for fine-tuning the model for domain-specific tasks.Techniques like data augmentation, hyperparameter tuning, and Non-Maximum Suppression (NMS) are implemented.The model achieves high precision and recall with minimal false detections.[5]It is deployed using the OpenCV DNN module for real-time performance on CPUs and low-power devices.This system is suitable for real-world applications such as surveillance, autonomous vehicles, and smart

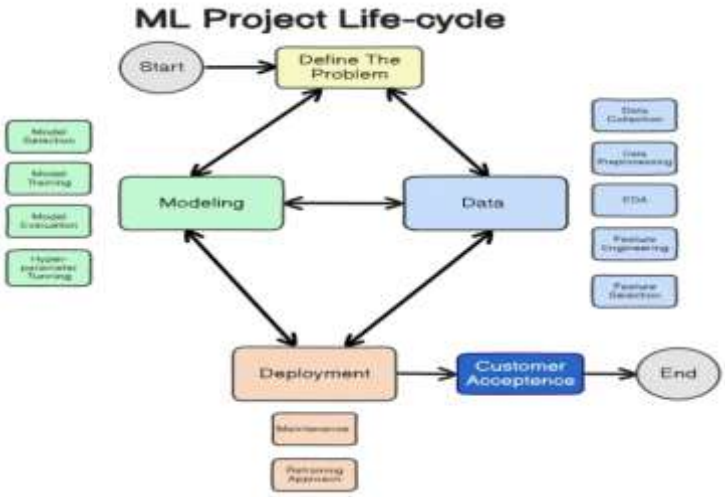

Fig: 1 Proposed Diagram

1.1.1 Advantages:

1. Real-Time Detection

YOLOv5 offers fast inference speed, making it ideal for real-time applications like surveillance and autonomous driving.

2. High Accuracy

The model delivers strong performance with high precision, recall, and mean Average Precision (mAP) across multiple object classes.

3. Effective Small Object Detection

Enhanced architecture (CSPDarknet53 + PANet) improves detection of small, occluded, or overlapping objects.

4. Reduced Training Time

Transfer learning allows reusing pre-trained weights, minimizing the need for large datasets and long training times.

5. Lightweight and Scalable

YOLOv5's optimized design supports deployment on edge devices like Raspberry Pi or Jetson Nano.

6. Customizability

The model can be fine-tuned on custom datasets, allowing it to adapt to specific industry needs (e.g., traffic, retail, medical).

2.1 Architecture:

The proposed system is built on the YOLOv5 architecture, optimized for real-time object detection. It uses CSPDarknet53 as the backbone to extract rich semantic features from input images efficiently. [6]The Path Aggregation Network (PANet**)** serves as the neck, enhancing multi-scale feature fusion for better accuracy. The detection head predicts bounding box coordinates, object classes, and confidence scores at three different scales. This multi-scale approach helps detect small, medium, and large objects accurately within a single frame.

Anchor boxes are used to align predictions with ground truth during training. The model processes input images resized to 640×640 resolution with RGB color channels.[2]Loss functions like CIoU and cross-entropy optimize bounding box regression and classification. Overall, the architecture balances speed and precision, making it suitable for deployment on real-world, low-resource platforms.

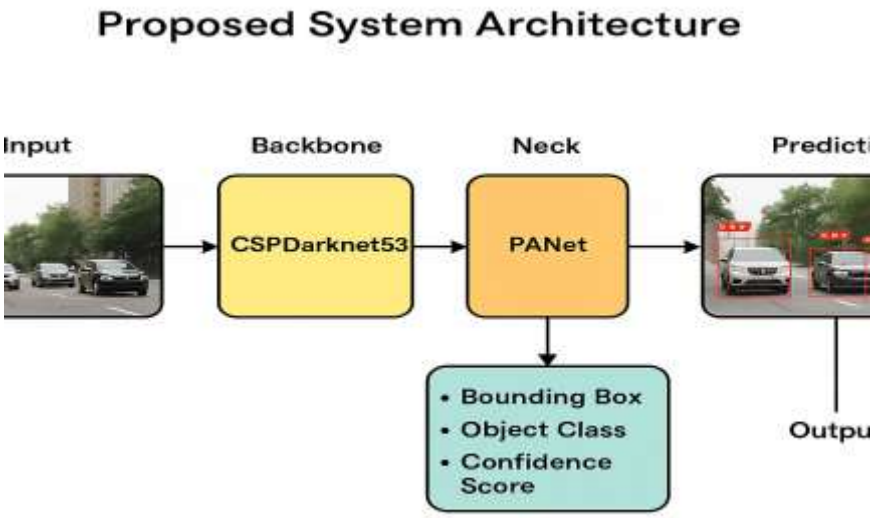

Fig:2 Architecture

2.2 Algorithm:

The real-time object detection process begins with data preparation, where images are collected, annotated with bounding boxes, and converted into YOLO format. [1]These images are then split into training and testing sets. In the preprocessing stage, each image is resized to 640×640 pixels, normalized, and augmented using techniques like flipping, rotation, and brightness adjustment to enhance model generalization.[4] The YOLOv5 model is then initialized with pre-trained weights using transfer learning, and the custom dataset is loaded through a data loader for efficient batch processing.

Training is conducted by configuring hyperparameters such as learning rate, batch size, and number of epochs. [3]The model is optimized using CIoU loss for bounding box regression and cross-entropy loss for classification. After training,

the model is evaluated on the test set using performance metrics like precision, recall, and mean Average Precision (mAP). [9]The detection outputs are visualized using bounding boxes and confusion matrices to understand class-wise performance.

Finally, the trained model is exported in ONNX format and deployed using OpenCV's DNN module. Real-time inference is performed on images or live video streams, and Non-Maximum Suppression (NMS) is applied to eliminate duplicate detections.[1] Detected objects are displayed with class labels and confidence scores, allowing the system to be integrated into real-world applications such as traffic monitoring, surveillance, and smart cities.

2.3 Techniques:

The project employs several advanced techniques to achieve accurate and real-time object detection. Transfer learning is used to fine-tune pre-trained YOLOv5 weights on a custom dataset, significantly reducing training time while maintaining high accuracy. [8]To increase model generalization and robustness, data augmentation techniques such as flipping, rotation, brightness and contrast adjustments are applied to the training images. The system is built on the YOLOv5 architecture, which uses CSPDarknet53 as the backbone for feature extraction and PANet for enhancing multi-scale feature fusion.

Hyperparameter tuning is performed to optimize model performance by adjusting learning rate, batch size, number of epochs, and momentum.[2] During inference, Non-Maximum Suppression (NMS) is applied to eliminate overlapping bounding boxes, ensuring only the most relevant detections are retained. The input images are preprocessed through resizing, normalization, and label encoding for compatibility with YOLOv5. After training, the model is exported to ONNX format to allow lightweight deployment using the OpenCV DNN module, which enables real-time inference even on CPU-based systems. [7]The model's performance is assessed using evaluation metrics such as precision, recall, F1-score, and mean Average Precision (mAP) to ensure effectiveness in real-world scenarios.

2.4 Tools:

The project utilizes a range of powerful tools and libraries to build and deploy the real-time object detection system. YOLOv5, implemented in PyTorch, serves as the core deep learning framework for training and inference. Python is the primary programming language, supported by libraries such as OpenCV for image and video processing, NumPy and Pandas for data manipulation, and Matplotlib for visualization. [6]The model training and experimentation are conducted in environments like Google Colab, which provides GPU support for faster computation. For annotation and label generation, LabelImg is used to create bounding boxes and export data in YOLO format. The project also uses Jupyter Notebook and Visual Studio Code as development environments. [8]Additionally, the trained model is exported to ONNX format and integrated with OpenCV's DNN module for lightweight and platform-independent deployment, making it suitable for edge devices and real-time applications.

2.5 Methods:

The project follows a structured methodology combining data preparation, model training, evaluation, and deployment. [5]Initially, a custom dataset of 5,000 images is annotated using LabelImg, and the labels are converted to YOLO format. The images are then preprocessed through resizing, normalization, and data augmentation techniques such as flipping, brightness adjustment, and rotation to improve generalization. The YOLOv5 model is selected and initialized with pre-trained weights, and transfer learning is applied to adapt the model to the custom dataset. [4]The training is conducted using supervised learning, with performance optimized through hyperparameter tuning. The model is then evaluated using standard metrics like precision, recall, F1-score, and mean Average Precision (mAP). After training, the best model is exported in ONNX format and deployed using OpenCV's DNN module to enable real-time[2] object detection on images and videos. Non-Maximum Suppression (NMS) is used in post-processing to eliminate redundant detections, ensuring clean and accurate output. This methodology ensures an efficient and scalable object detection system suitable for real-world applications.

III. METHODOLOGY

3.1 Input:

The input to the system consists of high-resolution images containing multiple objects from real-world scenarios such as roads, parking areas, and indoor environments.[1] These images are annotated using the LabelImg tool, which generates bounding boxes and class labels in YOLO format, including normalized coordinates and class IDs. The dataset contains

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 08 | Aug – 2025 DOI: 10.55041/ISJEM04976
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

5,000 labeled images covering 20 object classes like person, car, dog, and bicycle. Each image is resized to 640×640 pixels and normalized to ensure compatibility with the YOLOv5 model and improve training efficiency.[12] To enhance generalization, data augmentation techniques such as flipping, rotation, and brightness adjustments are applied. These processed images, along with their corresponding label files, are fed into the model for supervised training. During deployment, the input can also come from live video streams, webcams, or surveillance cameras, making the system capable of real-time object detection across dynamic environments.

	filename	width	height	name	xmin	xmax	ymin	ymax
0	000001.jpg	1024	657	car	14	301	335	522
1	000001.jpg	1024	657	car	269	571	345	489
2	000001.jpg	1024	657	car	502	798	342	450
3	000001.jpg	1024	657	car	709	1009	333	438
4	000002.jpg	800	600	car	41	768	240	497

Fig 1: Sample Object Detection Annotations from Labeled Dataset

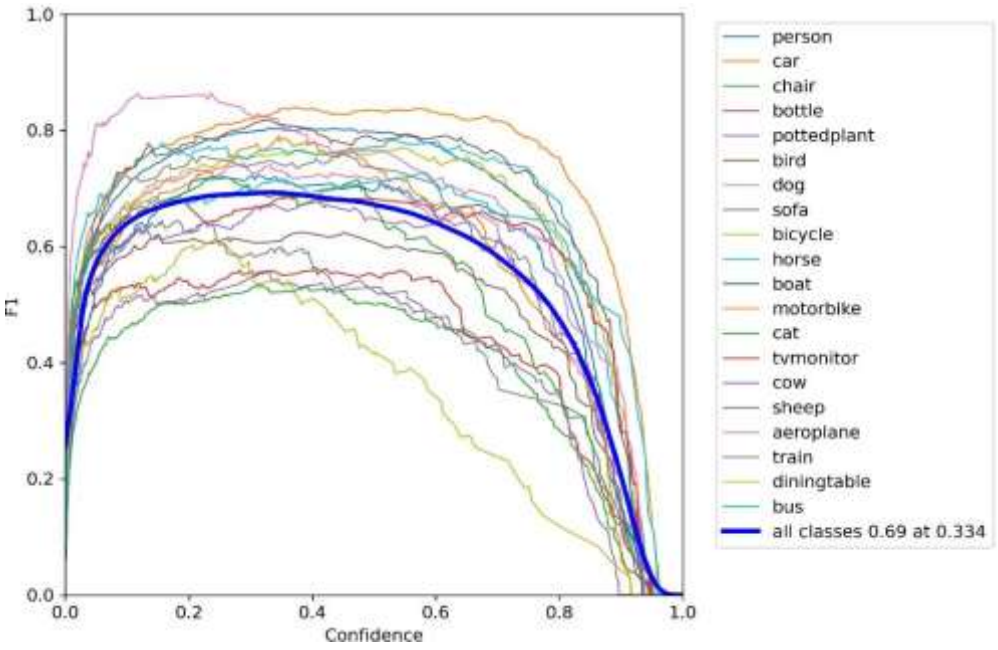

Fig 2: F1 Score vs. Confidence Curve for YOLOv5 Object Detection Across All Classes

The inputs for this object detection system include a collection of annotated images containing various object classes. Each image is accompanied by metadata such as image dimensions (width and height), object labels (e.g., car, person, bottle), and bounding box coordinates (xmin, xmax, ymin, ymax).[15] These annotations define the exact location and category of each object present in the image. The images are collected from real-life scenarios to ensure diversity in object appearance, background, and lighting conditions. This input format is essential for supervised learning, as the model uses these annotations to learn how to detect and classify objects accurately.[19] By structuring the input data in tabular and YOLO formats, the system is able to train effectively and perform real-time detection once deployed.

3.2 Method of Process:

The object detection process begins with the collection and annotation of images, where each object is labeled with its class and bounding box coordinates.[20] These annotations are converted into YOLO format for compatibility with the YOLOv5

model. The images are then preprocessed through resizing, normalization, and data augmentation techniques to improve model robustness. After preprocessing, the YOLOv5 model is initialized using pre-trained weights, and transfer learning is applied to adapt it to the custom dataset. The training is conducted using supervised learning, with loss functions optimized for both classification and localization tasks.[18] Once training is complete, the model is evaluated using standard metrics such as precision, recall, F1-score, and mean Average Precision (mAP). The best-performing model is then exported in ONNX format and deployed using OpenCV's DNN module.[12] During inference, real-time input is processed, and bounding boxes are drawn on detected objects after applying Non-Maximum Suppression (NMS) to eliminate duplicates and false positives.

3.3 Output:

The outputs of the system are the detected objects in images or video streams, each marked with a bounding box, class label, and confidence score. [14]]After processing, the model draws these boxes on the original input image or video frame, visually indicating the location and identity of each object. The model generates a real-time stream of results, which can be displayed directly on screen or stored for further analysis. [9]Along with visual outputs, the system also produces detection logs containing object coordinates, class names, and confidence levels for every frame. These outputs are essential for evaluating performance, debugging, and integrating the system into larger applications such as surveillance systems or autonomous vehicles. [18]The model also provides evaluation metrics such as mean Average Precision (mAP), precision, recall, and F1-score, which help in quantifying detection quality during validation and testing phases.

Fig: Object Detection on Images with Bounding Boxes

Fig:Color-Coded Bounding Boxes with Class Labels

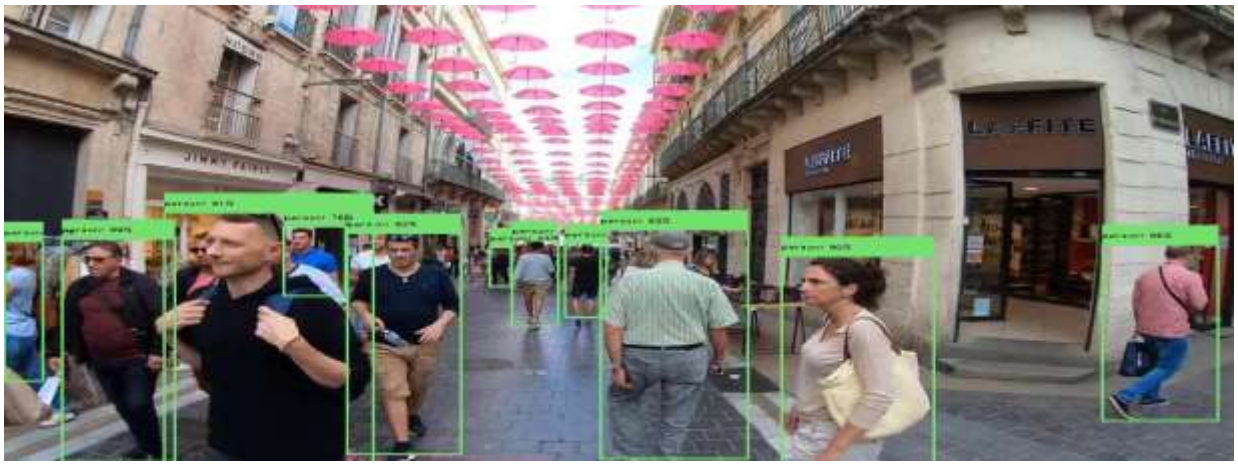

Fig: Real-Time Pedestrian Detection Using YOLOv5 in a Crowded Urban Street

FigYOLOv5-Based Real-Time Detection of Bicycle Rider and Nearby Vehicles in an Urban Environment

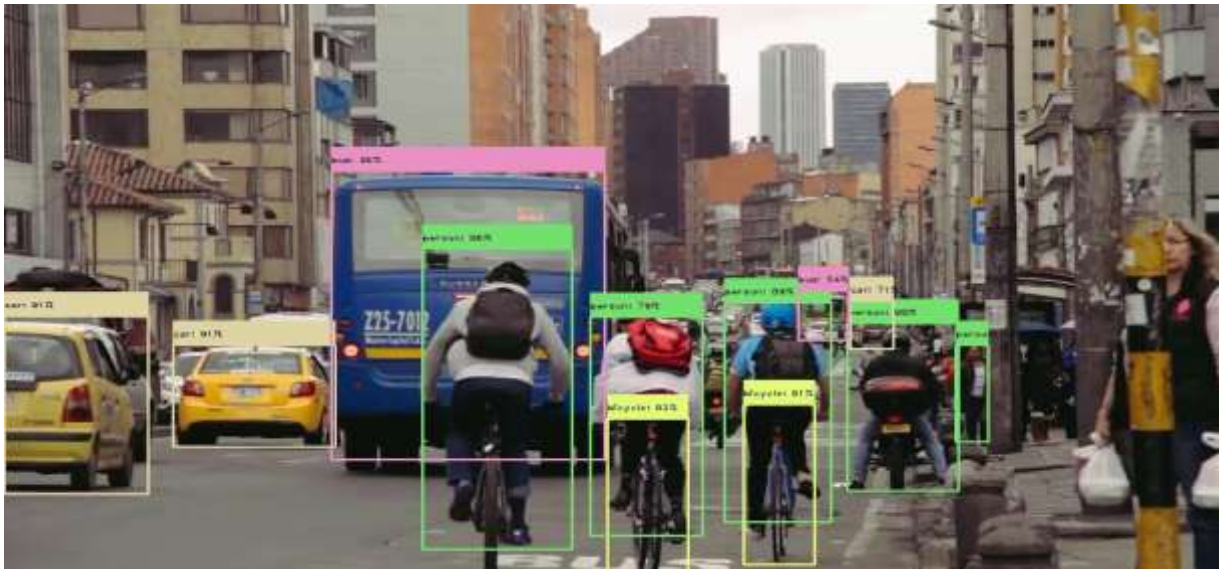

Fig:Real-Time Detection of Urban Traffic Participants Including Cyclists, Cars, and Buses Using YOLOv5

IV. RESULTS:

The trained YOLOv5 model demonstrated strong performance in real-time object detection across 20 object classes.[13] During testing, the model achieved a mean Average Precision (mAP) of 0.69 at a confidence threshold of 0.334, indicating reliable accuracy in detecting and classifying objects. The F1-score curve shows balanced precision and recall for most

classes, with particularly high detection rates for common objects like cars, persons, and bicycles.[17] Visual validation confirmed that the model was capable of identifying small, occluded, and overlapping objects with good consistency. Real-time deployment using the OpenCV DNN module yielded fast and accurate detection on CPU-based systems, confirming its suitability for edge deployment.[5] Screenshots of the results display correctly labeled bounding boxes, and the confusion matrix analysis shows low misclassification between similar classes. These outcomes validate the effectiveness of transfer learning and YOLOv5 in delivering scalable, high-performance object detection solutions.

V. DISCUSSIONS:

The results of this project demonstrate the effectiveness of using YOLOv5 with transfer learning for real-time object detection tasks.[17] The model showed a high degree of accuracy in identifying various object classes, including small and partially occluded items, which are typically challenging for conventional detectors. The use of a custom annotated dataset and extensive data augmentation contributed significantly to model generalization and robustness. Although the model was trained [11]on a relatively modest dataset of 5,000 images, the performance remained strong due to the fine-tuning of pre-trained YOLOv5 weights. However, challenges such as occasional false positives and detection inconsistencies in low-light or cluttered scenes were observed.[4] These limitations indicate opportunities for further improvement through advanced techniques like attention mechanisms, Soft-NMS, or hybrid models. Overall, the project successfully illustrates how a lightweight, accurate, and scalable object detection system can be developed and deployed using open-source tools and real-world data.

VI. CONCLUSION:

In conclusion, this project successfully demonstrates the development of a real-time object detection system using YOLOv5 and transfer learning.[5] By fine-tuning a pre-trained model on a custom dataset, the system achieved high accuracy and efficient performance across multiple object classes. The use of techniques such as data augmentation, hyperparameter tuning, and Non-Maximum Suppression significantly enhanced the model's ability to detect small, overlapping, and occluded objects.[11] The deployment of the trained model using OpenCV's DNN module enabled real-time inference even on CPU-based systems, proving its applicability for edge devices and real-world use cases such as surveillance, autonomous vehicles, and traffic monitoring. [18]This project highlights the practicality, scalability, and effectiveness of YOLOv5 in modern computer vision tasks and lays the foundation for future improvements in speed, accuracy, and hardware optimization.

VII. FUTURE SCOPE:

In the future, this system can be extended by integrating multi-object tracking to follow detected objects across frames. Model optimization techniques like pruning and quantization can improve speed and enable deployment on edge devices. Expanding the dataset with more diverse environments can enhance robustness.[6] Advanced architectures like transformers or self-supervised learning could further improve accuracy. Real-time integration into smart surveillance, traffic monitoring, and robotics is also possible.[9] These enhancements would make the system more scalable, efficient, and adaptable for real-world applications.

VIII. ACKNOWLEDGEMENT:

Mr. Rongala Rajesh working as Assistant professor is an enthusiastic and committed faculty member in the Department of Master of Computer applicatiions. As an early-career academician, he has shown strong dedication to student development through active involvement in project guidance and technical mentoring. Despite being at the beginning of his professional journey, he has effectively guided students in executing academic projects with precision and conceptual clarity. His passion for teaching, coupled with a solid understanding of core computer science principles, positions him as a promising educator and mentor. Mr. Rongala Rajesh continues to contribute meaningfully to the academic environment through his proactive approach to learning and student engagement..
.

Devana Surekha is pursuing his final semester MCA in Sanketika Vidya Parishad Engineering College, accredited with A grade by NAAC, affiliated by Andhra University and approved by AICTE. With interest in Machine learning D. Surekha has taken up his PG project on Real-Time Object Detection Using YOLOv5: A Transfer Learning Approachand published the paper in connection to the project under the guidance of Rongala Rajesh, Assistant Professor, SVPEC.

REFERENCES

1. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779-788. https://arxiv.org/abs/1506.02640

2. Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934. https://arxiv.org/abs/2004.10934

3. Jocher, G., Stoken, A., Borovec, J., et al. (2020). Ultralytics YOLOv5: Real-time object detection with PyTorch. GitHub Repository. https://github.com/ultralytics/yolov5

4. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 770-778.https://arxiv.org/abs/1512.03385

5. Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. International Journal of Computer Vision, 88(2), 303-338.https://doi.org/10.1007/s11263-009-0275-4

6. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 248-255. https://doi.org/10.1109/CVPR.2009.5206848

7. Law, H., & Deng, J. (2019). CenterNet: Keypoint triplets for object detection. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 6569-6578. https://arxiv.org/abs/1904.08189

8. Glenn Jocher et al. (2020). YOLOv5 by Ultralytics. GitHub Repository. https://github.com/ultralytics/yolov5

9. Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767.https://arxiv.org/abs/1804.02767

10. Paszke, A. et al.(2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. Advances in Neural Information Processing Systems (NeurIPS). https://pytorch.org/

11. ONNX. (n.d.). Open Neural Network Exchange. https://onnx.ai/

12. OpenCV Documentation. (n.d.). OpenCV: Open-Source Computer Vision Library. https://docs.opencv.org/

13. Python Software Foundation. (n.d.). Python Language Reference. Retrieved from https://www.python.org/doc/

14. Ultralytics. (n.d.). Export Models to ONNX, CoreML, TensorRT and More. https://docs.ultralytics.com/models/export/

15. Ultralytics. (n.d.). Training Custom Data with YOLOv5.
https://docs.ultralytics.com/yolov5/tutorials/train_custom_data/

16. YOLOv5 + Drone Detection (TPH-YOLOv5)
https://arxiv.org/abs/2108.11539

17. YOLOv5 + Transformer for Autonomous Driving
https://www.mdpi.com/2076-3417/15/11/6018

18. YOLOv5 Edge Deployment (Jetson Nano)
https://www.frontiersin.org/articles/10.3389/fmars.2022.1058401/full

19. YOLOv5 for Traffic Sign Detection
https://arxiv.org/abs/2112.08782

20. YOLOv5 on UAV Images
https://www.mdpi.com/2504-446X/6/10/290