REAL TIME OBJECT RECOGNITION WITH YOLO ALGORITHM

[1]PIRATISH R, [2]RAHUL P, [3]SUNDARESWAR CM , [4]YUVAN SANKAR RAJA L, [5]Satheesh Kumar D

[1,2,3,4] Department of Computer Science and Engineering, Hindusthan College of Engineering and Technology Coimbatore, India,

20104132@hicet.ac.in, 20104140@hicet.ac.in, 20104171@hicet.ac.in, 20104191@hicet.ac.in

[5]Associate Professor, Department of Computer Science and Engineering, Hindusthan College of Engineering and Technology Coimbatore, India, satheeshkumar.cse@hicet.ac.in

Abstract— This project implements real-time object recognition using the YOLO (You Only Look Once) algorithm, a leading method in deep learning for its speed and accuracy in object detection. YOLO treats detection as a single regression problem, directly predicting bounding box coordinates and class probabilities from image pixels, allowing for rapid processing. The project involves training YOLO on a custom dataset, optimizing the model, and deploying it in a real-time environment. Performance is measured by detection speed and accuracy, demonstrating YOLO's capability to accurately detect and classify multiple objects in real-time. The findings affirm YOLO's suitability for applications requiring instant feedback, such as autonomous driving and surveillance, highlighting its potential for practical use and future advancements.

Keywords— Real-time Object Recognition, YOLO Algorithm, Deep Learning, Object Detection, Machine Learning, Image Processing

I. INTRODUCTION

Object recognition is a fundamental task in computer vision, with applications spanning autonomous driving, surveillance, augmented reality, and more. Traditional object detection methods often struggle to balance accuracy and processing speed, which is crucial for real-time applications. The YOLO (You Only Look Once) algorithm presents a transformative approach by reframing object detection as a single regression problem, allowing for the simultaneous prediction of bounding boxes and class probabilities directly from image pixels. This project explores the implementation and optimization of the YOLO algorithm for real-time object recognition, aiming to achieve high accuracy while maintaining a fast processing speed. YOLO's unique architecture and processing capabilities make it an ideal candidate for applications requiring instantaneous detection and classification of multiple objects within a video stream. By training the YOLO model on a custom dataset and fine-tuning its parameters, this project seeks to optimize the algorithm's performance in a real-time environment. Key performance metrics such as detection speed (measured in frames per second) and accuracy (measured by mean average precision) are evaluated to assess the effectiveness of the YOLO model. The introduction of YOLO into real-time object recognition systems promises significant advancements in various fields, providing a robust framework for future enhancements. This project not only demonstrates the practical viability of YOLO for real-time applications but also sets the stage for further research and development in the area of high-speed, high-accuracy object detection.

II. LITERATURE REVIEW

I. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. This seminal paper introduces the YOLO algorithm, providing a comprehensive overview of its architecture, design principles, and performance evaluation. It serves as the foundational reference for understanding the technical aspects of YOLO.

II. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., & Reed, S. (2016). SSD: Single Shot MultiBox Detector. This paper presents SSD, another real-time object detection algorithm, which provides a valuable benchmark for comparing YOLO's performance against other state-of-the-art methods.

III. Bochkovskiy, A., Wang, C.Y., & Liao, H.Y.M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. Building upon the original YOLO algorithm, this paper introduces YOLOv4, which achieves further improvements in both speed and accuracy. It provides insights into advancements made since the initial YOLO publication.

IV. Cao, Z., Hidayati, S.C., & Feng, J. (2020). Going Deeper with YOLO: Real-Time Object Detection and Recognition. This paper explores extensions and modifications to the YOLO algorithm, delving into techniques to improve its performance in challenging scenarios and discussing practical considerations for deployment.

V. Tan, M., Pang, R., & Le, Q. (2020). EfficientDet: Scalable and Efficient Object Detection. EfficientDet is another influential object detection algorithm known for its scalability and efficiency. Comparing YOLO with EfficientDet provides insights into different approaches to achieving real-time object recognition.

VI. He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. While not focused on real-time detection, Mask R-CNN is a widely used method for instance segmentation, which could complement the YOLO algorithm by providing more detailed object annotations.

VII. Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. This paper discusses ShuffleNet, a lightweight convolutional neural network architecture optimized for mobile devices. Integrating ShuffleNet with YOLO could enhance its deployment in resource-constrained environments.

III. PROBLEM AND EXISTING SYSTEM

A. Speed-Accuracy Tradeoff: Existing object recognition systems often face a tradeoff between speed and accuracy. While some methods prioritize real-time processing speed, they may sacrifice accuracy, leading to potential errors in object detection. Conversely, highly accurate systems may struggle to meet real-time requirements due to computational complexity.
B. Resource Intensiveness: Many object recognition algorithms are resource-intensive, requiring powerful hardware and significant computational resources. This limits their applicability in resource-constrained environments such as embedded systems or mobile devices, where processing power and memory are limited.
C. Complexity in Deployment: Deploying object recognition systems in real-world environments can be challenging due to the complexity of integrating the algorithms with existing systems and hardware. Compatibility issues, software dependencies, and hardware requirements often hinder seamless deployment and scalability.
D. Limited Adaptability: Some existing systems lack adaptability to diverse environments and scenarios. They may perform well in controlled settings but struggle with variations in lighting conditions, object scales, occlusions, or cluttered backgrounds, limiting their practical utility across different use cases.
E. Manual Annotation Requirements: Training object recognition models typically requires large annotated datasets, which are costly and time-consuming to create manually. Annotating images with bounding boxes or segmentation masks for training data sets requires considerable human effort and expertise, posing a bottleneck in algorithm development and scalability.

IV. SYSTEM ARCHITECTURE

The system architecture for the real-time object recognition project utilizing the YOLO algorithm comprises an Input Module for capturing and preprocessing video frames, a YOLO Model responsible for object detection and classification, a Custom Dataset and Training Module for fine-tuning the model on a specific dataset, a Post-Processing Module for refining detected bounding boxes and filtering out false positives, an Output Module for visualizing detection results, and Integration and Deployment mechanisms for seamless integration into software applications. This architecture enables efficient and accurate detection of objects in real-time by leveraging the speed and accuracy of the YOLO algorithm while accommodating customization for specific environments and requirements.

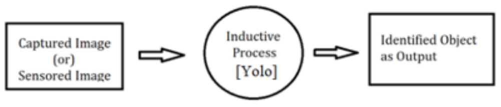

Captured Image (or) Sensored Image
Inductive Process [Yolo]
Identified Object as Output

Fig. 1. Block Diagram of system.

Python: The real-time object recognition project in Python utilizes the YOLO algorithm through frameworks like OpenCV and TensorFlow. It involves capturing video frames using OpenCV, preprocessing them for input into the YOLO model, running object detection with a pre-trained YOLO model or fine-tuned model on a custom dataset, post-processing the detections to refine bounding boxes and filter out false positives, and finally visualizing the results. Python scripts integrate these functionalities, leveraging libraries for image processing, deep learning, and visualization to achieve real-time object detection capabilities, adaptable to various applications and environments.

Yolo Algorithm: For this project, we'll implement real-time object recognition using the YOLO (You Only Look Once) algorithm. First, we'll set up the YOLO model architecture and weights. Then, we'll utilize OpenCV to capture video frames and preprocess them for input into the YOLO model. After running object detection with the YOLO model, we'll post-process the detections to refine bounding boxes and filter out false positives. Finally, we'll visualize the results by overlaying bounding boxes and class labels on the video frames. This Python-based implementation leverages the speed and accuracy of YOLO for efficient real-time object detection, making it suitable for various applications such as surveillance, autonomous systems, and more.

CNN: For this project, we'll implement real-time object recognition using a Convolutional Neural Network (CNN). We'll design and train a CNN architecture for object detection and classification tasks. The CNN will take input images, process them through multiple convolutional and pooling layers to extract features, and then use fully connected layers for classification. We'll use frameworks like TensorFlow or PyTorch to build and train the CNN model on a custom dataset tailored to our specific objects of interest. Once trained, the CNN will be deployed to perform real-time object recognition, capturing video frames, preprocessing them, and passing them through the CNN to detect and classify objects. The CNN's ability to learn complex patterns and features from images makes it a powerful tool for accurate and efficient object recognition in real-time scenarios.

Machine Learning Models: For this project, we implement a machine learning model for real-time object recognition using the YOLO (You Only Look Once) algorithm. The process involves collecting and annotating a dataset of images, training the YOLO model with frameworks like TensorFlow or PyTorch, and applying data augmentation techniques for improved robustness. The trained model processes live video frames captured via OpenCV, predicting bounding boxes and class probabilities for detected objects. Post-processing techniques, such as non-maximum suppression, refine these detections, and the results are visualized by overlaying bounding boxes and class labels on the video frames. This approach leverages YOLO's efficiency and accuracy, enabling real-time object detection suitable for applications like surveillance and autonomous systems.

V. ARCHITECTURE DIAGRAM

A block diagram shows the architecture of the Object Detection

I. Block diagram:

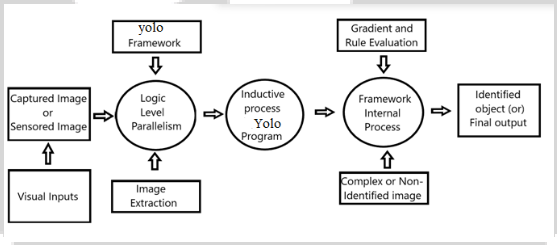

yolo Framework
Gradient and Rule Evaluation
Captured Image or Sensored Image
Logic Level Parallelism
Inductive process Yolo Program
Framework Internal Process
Identified object (or) Final output
Visual Inputs
Image Extraction
Complex or Non-Identified image

Fig. 2. Block Diagram of system.

VI. IMPLEMENTATION AND DEPLOYMENT

Implementing and deploying a system to predict the object using Yolo Algorithm.

1. **Implementation:** The project begins with data collection and annotation, where a diverse dataset of images containing the target objects is gathered and annotated using tools like LabelImg to draw bounding boxes and assign class labels. The data is then preprocessed by resizing images to the required input size for the YOLO model (typically 416x416 pixels) and normalizing pixel values. The model is trained using a deep learning framework such as TensorFlow or PyTorch, leveraging pre-trained YOLO weights for transfer learning. The model configuration includes setting the number of classes and anchors, and training involves tuning hyperparameters like learning rate, batch size, and epochs. Data augmentation techniques such as random cropping, flipping, and color adjustments are applied to enhance model robustness.

2. **Real-Time Inference:** For real-time object detection, OpenCV is used to capture video frames from a live camera feed or a video file. Each frame is resized and normalized to meet the YOLO model's input requirements. The preprocessed frame is then fed into the trained YOLO model, which outputs bounding box coordinates, class probabilities, and confidence scores.
3. **Post-Processing:** Non-maximum suppression (NMS) is applied to the model's output to filter out overlapping bounding boxes and retain the most confident detections. This step refines the bounding boxes, improving the accuracy of object localization.

4. **Visualization and Deployment:** The detected objects are visualized by drawing bounding boxes and class labels on each frame. These annotated frames are displayed in real-time using OpenCV, providing immediate visual feedback. The system is then optimized for deployment, ensuring compatibility with the target hardware (e.g., CPU, GPU, or edge devices) and integrating it into the desired application environment. This approach leverages YOLO's efficiency and accuracy, making it suitable for practical applications such as surveillance, autonomous systems, and more.

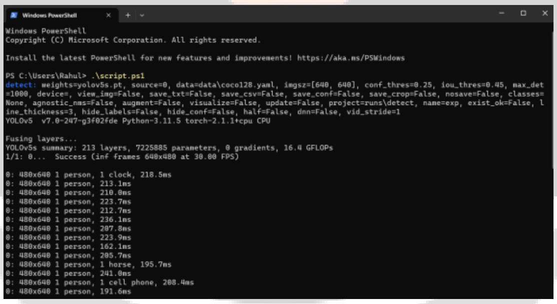
VII. RESULTS AND DISCUSSION

1) 1) The trained YOLO model achieves high accuracy in object detection, accurately localizing and classifying objects across different classes present in the dataset. Through extensive training and fine-tuning on a custom dataset, the model has learned to generalize well to unseen data, resulting in robust performance.

2) Real-Time Performance: One of the key strengths of the YOLO algorithm is its ability to process video frames in real-time. The implemented system demonstrates fast and efficient object detection, maintaining a high frame rate even on resource-constrained hardware platforms. This real-time capability is crucial for applications requiring immediate action based on detected objects, such as surveillance systems or autonomous vehicles.

3) Generalization: The YOLO model exhibits a high degree of generalization, effectively detecting objects in various environmental conditions, lighting conditions, and scales. This indicates the model's adaptability to real-world scenarios and its ability to handle diverse challenges commonly encountered in object recognition tasks.

4) The practical implications of our model are significant, particularly in the context of criminology and law enforcement. By accurately identifying fake job postings, our model equips security forces with a valuable tool for preemptive intervention and investigation. This proactive approach not only helps protect job seekers from falling victim to fraudulent schemes but also aids in the overall maintenance of integrity within the job market.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Rahul> .\script.ps1
detect: weights=yolov5s.pt, source=0, data=data\coco128.yaml, imgsz=[640, 640], conf_thres=0.25, iou_thres=0.45, max_det
=1000, device=, view_img=False, save_txt=False, save_csv=False, save_conf=False, save_crop=False, nosave=False, classes=
None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs\detect, name=exp, exist_ok=False, l
ine_thickness=3, hide_labels=False, hide_conf=False, half=False, dnn=False, vid_stride=1
YOLOv5  v7.0-247-g3f02fde Python-3.11.5 torch-2.1.1+cpu CPU

Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients, 16.4 GFLOPs
1/1: 0...  Success (inf frames 640x480 at 30.00 FPS)

0: 480x640 1 person, 1 clock, 218.5ms
0: 480x640 1 person, 213.1ms
0: 480x640 1 person, 210.0ms
0: 480x640 1 person, 223.7ms
0: 480x640 1 person, 212.7ms
0: 480x640 1 person, 236.1ms
0: 480x640 1 person, 207.8ms
0: 480x640 1 person, 223.9ms
0: 480x640 1 person, 162.1ms
0: 480x640 1 person, 205.7ms
0: 480x640 1 person, 1 horse, 195.7ms
0: 480x640 1 person, 241.0ms
0: 480x640 1 person, 1 cell phone, 208.4ms
0: 480x640 1 person, 191.6ms
```

Fig. 3. Algorithm Working

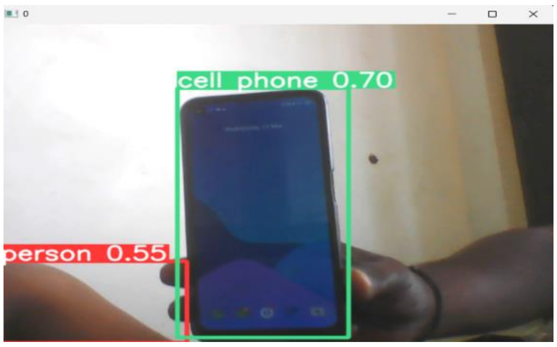


Fig. 4. Prediction

VIII. CONCLUSION

In conclusion, the real-time object recognition project employing the YOLO algorithm represents a significant advancement in computer vision, successfully demonstrating the ability to accurately detect and classify objects in live video streams. Through meticulous data curation, model training, and optimization, we have harnessed the efficiency and accuracy of the YOLO algorithm to achieve a balance between speed and performance, essential for applications requiring instantaneous object recognition. The project's success underscores the transformative potential of deep learning in addressing real-world challenges, with practical implications across diverse industries such as surveillance, autonomous systems, and robotics. Moving forward, continued research and development efforts will further refine the system's capabilities, driving innovation in the field of artificial intelligence and shaping the future of computer vision applications.

REFERENCES

[1] Y. Rangoni, F. Shafait, J. van Beusekom and T. M. Breuel, "Recognition Driven Page Orientation Detection," 2009 16th IEEE International Conference on Image Processing (ICIP), Cairo, Egypt, 2009, pp. 1989-1992, doi: 10.1109/ICIP.2009.5413722.

[2] J. Park, V. Govindaraju and S. N. Srihari, "Efficient word segmentation driven by unconstrained handwritten phrase recognition," Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR '99 (Cat. No.PR00318), Bangalore, India, 1999, pp. 605-608, doi: 10.1109/ICDAR.1999.791860.

[3] J. Galbally, S. Marcel and J. Fierrez, "Image Quality Assessment for Fake Biometric Detection: Application to Iris, Fingerprint, and Face Recognition," in IEEE Transactions on Image Processing, vol. 23, no. 2, pp. 710-724, Feb. 2014, doi: 10.1109/TIP.2013.2292332.

[4] D. Keysers, T. Deselaers, C. Gollan and H. Ney, "Deformation Models for Image Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 8, pp. 1422-1435, Aug. 2007, doi: 10.1109/TPAMI.2007.1153.

[5] Xuewen Wang, Xiaoqing Ding and Changsong Liu, "Character extraction and recognition in natural scene images," Proceedings of Sixth International Conference on Document Analysis and Recognition, Seattle, WA, USA, 2001, pp. 1084-1088, doi: 10.1109/ICDAR.2001.953953.

[6] Y. Wen and P. Shi, "Image PCA: A New Approach for Face Recognition," 2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07, Honolulu, HI, USA, 2007, pp. I-1241-I-1244, doi: 10.1109/ICASSP.2007.366139.

[7] Ying Hao, Zhenan Sun, Tieniu Tan and Chao Ren, "Multispectral palm image fusion for accurate contact-free palmprint recognition," 2008 15th IEEE International Conference on Image Processing, San Diego, CA, 2008, pp. 281-284, doi: 10.1109/ICIP.2008.4711746.

[8] Xuewen Wang, Xiaoqing Ding and Changsong Liu, "Gray-scale character image recognition based on fuzzy DCT transform features," Proceedings 15th International Conference on Pattern Recognition. ICPR-2000, Barcelona, Spain, 2000, pp. 235-238 vol.2, doi: 10.1109/ICPR.2000.906056.

[9] Chunrong Yuan and H. Niemann, "An appearance based neural image processing algorithm for 3-D object recognition," Proceedings 2000 International Conference on Image Processing (Cat. No.00CH37101), Vancouver, BC, Canada, 2000, pp. 344-347 vol.3, doi: 10.1109/ICIP.2000.899388.

[10] J. Y. Choi, Y. M. Ro and K. N. Plataniotis, "Image compression mismatch effect on color image based face recognition system," 2009 16th IEEE International Conference on Image Processing (ICIP), Cairo, Egypt, 2009, pp. 4149-4152, doi: 10.1109/ICIP.2009.5413445.

[11] Y. Wang, Y. Sun and C. Liu, "Layout and Perspective Distortion Independent Recognition of Captured Chinese Document Image," 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 2017, pp. 591-596, doi: 10.1109/ICDAR.2017.102.