Real-Time Object Detection System Using YOLO and Raspberry Pi

Dr.Preethi S[*a], Rakshith Manjunatha [b], Renold Stephen [b], Sam Melwin [b], Vishnu Prasad [b]
[*a] a Professor and HOD, Dept. of ISE
[b] UG Students, Dept. of ISE
[a,b*] *Department of ISE, Cambridge Institute of Technology (Autonomous), K R Puram, Bengaluru, India*

***Abstract*— This paper presents a real-time object detection system designed to assist visually impaired individuals, leveraging the capabilities of YOLOv5 and the Raspberry Pi 4B. The system integrates a camera module for visual data acquisition, processed by the YOLOv5 algorithm to identify and classify objects in the user's surroundings. The Raspberry Pi 4B serves as the central processing unit, enabling efficient real-time analysis and minimizing latency. Complementary ultrasonic sensors are incorporated to provide distance information, enhancing spatial awareness and obstacle avoidance. The system aims to deliver timely auditory feedback, describing detected objects and their approximate distances, thereby improving navigation and situational awareness for blind users. The implementation focuses on optimizing YOLOv5 for the Raspberry Pi's computational constraints, ensuring a balance between detection accuracy and processing speed. Experimental results demonstrate the system's ability to accurately detect common objects in real-time, coupled with reliable distance measurement. This research contributes to the development of affordable and accessible assistive technologies for the visually impaired, promoting independent mobility and improved quality of life.**

Keywords: Object Detection, YOLOv5, Raspberry Pi, Assistive Technology, Visual Impairment.

I. INTRODUCTION

YOLO, or "You Only Look Once," is a pioneering technique to object recognition that has transformed the area of computer vision.

In contrast to conventional techniques that need to go over a picture several times in order to recognize objects, YOLO views object detection as a single regression problem. With the help of this novel architecture, the model can predict bounding boxes and class probabilities straight from complete images in a single assessment, greatly increasing processing performance.[1]. The architecture of YOLO is designed for high efficiency, making it especially appropriate for uses where performance in real time is crucial., such as autonomous driving, surveillance systems, and robotics. By leveraging deep learning techniques, YOLO achieves impressive accuracy while maintaining rapid inference times. Its capacity to simultaneously detect several items within a single frame sets it apart from other detection models, making it a popular option for both researchers and developers. The need for object detection in real time continues to grow across various industries, YOLO stands out as a powerful tool that meets both speed and accuracy requirements. The evolution of YOLO has addressed multiple challenges such as improving detection accuracy, reducing false positives, and optimizing computational efficiency. Each version incorporates architectural refinements, making YOLO a preferred choice for diverse real-world applications including surveillance, medical imaging, and autonomous navigation [4].

Firstly, YOLO is quite quick. A sophisticated process is not required since we define detection as a regression problem. At test time, we just apply our neural network to a fresh image in order to forecast detections. With no batch processing, our base network operates at 45 frames per second on a Titan X GPU, while our fast version operates at over 150 frames per second. With a latency of less than 25 milliseconds, we can handle streaming video in real-time. Moreover, YOLO outperforms other real-time systems in terms of mean average accuracy by more than double.

Image classification, which is defined as figuring out the class of the image, was one of the essential issues. The challenge of image localization, when one item is present in the picture and the system must predict its class and position within the image, is rather

challenging (a bounding box around the object).[1] The fact that objects discovery includes both identification and localisation makes it a more challenging challenge. In this instance, an image will be used as the system's input, and The result will be a bounding box that identifies the kind of object in each box and matches each object in the image. In order to achieve faster object identification and higher frames per second, we developed a system that consumes less computing power than the current methods [1,2]. Our model's algorithm locates a certain object by analyzing its appearance in a picture.

J. Redmon, "You Only Look Once: Unified, Real-Time Object Detection." (2016) In order to forecast bounding boxes and class probabilities at the same time, Redmon offers YOLO as a cohesive model that approaches object identification as a single regression issue. This model uses a straightforward architecture that only needs one forward neural network pass to enable high-speed detection. Each cell in the grid created by YOLO is tasked with predicting a bounding box and class probabilities. YOLO's remarkable speed and comparatively high accuracy for real-time applications are among its main features. However, the study also points out that the grid cells have trouble managing complicated backgrounds with numerous items and recognizing tiny things.

Farhadi, Ali, and Joseph Redmon. "YOLOv3: An Incremental Improvement." (2018) Redmon and Farhadi build upon the original YOLO architecture with YOLOv3, which aims to address limitations in accuracy and detection of small objects. YOLOv3 uses a more complex, deeper architecture with residual blocks and multi-scale prediction, which improves the ability to detect the objects of various sizes. It incorporates a Darknet-53 backbone, which enables higher accuracy in feature extraction. YOLOv3's multi-label classification also allows it to detect overlapping classes, a feature that earlier versions lacked. Although YOLOv3 sacrifices some speed compared to the previous YOLO, it represents a significant improvement in form of detection accuracy and flexibility.

Diwan, Tausif, G. Anirudh, and Jitendra V. Tembhurne. "Object Detection Using YOLO: Challenges, Architectural Successors, Datasets, and Applications." (2023)
This comprehensive review discusses the evolution of the YOLO architecture from its inception to recent advances. The authors analyze various versions of YOLO, focusing on architectural improvements across YOLO models, including YOLOv3, YOLOv4, and YOLOv5. The paper highlights the challenges YOLO faces, such as handling occlusions, detecting tiny objects and balancing detection speed with accuracy. Additionally, the review discusses diverse datasets and application domains where YOLO has been applied, from autonomous driving and surveillance to medical imaging. The authors suggest that while YOLO has made great progress, yet there are still obstacles to overcome and opportunities for further research to address complex, real-world scenarios.

Liu, Chengji, et al. "Object Detection on YOLO Network." (2018) Liu et al. analyze the YOLO framework’s effectiveness in different real-world environments and offer insights into optimizing YOLO’s architecture to improve performance. They propose modifications to the network's structure to better handle real-world images that may vary in scale, lighting, and orientation. The authors highlight YOLO's application in monitoring, surveillance, and quality control systems, showcasing its suitability for tasks requiring fast processing times. The paper also examines the accuracy-speed trade-offs, emphasizing scenarios where YOLO’s rapid detection can be maximally utilized despite occasional drops in accuracy for complex scenes.

Real-Time Speed: YOLO achieves real-time object detection speeds, processing images at up to 45 frames per second in its fastest versions, a major advantage in applications where speed is critical, such as autonomous vehicles, surveillance, and augmented reality.
A Tiny-ML technique built on Raspberry Pi improves real-time multiple item detection performance while consuming minimal power. The study investigated how lightweight neural networks can be optimised for all embedded devices, which is useful in resource-constrained contexts [6].
YOLOv3 improved the feature extraction technique by utilising a multi-scale detection strategy and residual connections, resulting in higher accuracy while preserving real-time speed. These modifications made

YOLOv3 appropriate for applications that require high precision and low latency [12].

The YOLO architecture combines classification and localisation tasks into a single neural network, resulting in faster processing and higher detection accuracy. Unlike standard object identification systems, YOLO does not rely on region proposal networks, which makes it substantially faster [13].

1. Single-Pass Detection: Unlike traditional approaches, YOLO does not rely on region proposals or exhaustive sliding windows to localize objects. Instead, it processes the full image at once, enabling it to anticipate several bounding boxes at once. and their associated class probabilities.
2. Unified Architecture: YOLO formulates object detection as a single, end-to-end neural network task, which simplifies training and makes deployment more straightforward than multi-stage methods. By implementing a unified, single-shot detection technique that greatly enhanced real-time performance, the original YOLO framework transformed object detection. YOLO reduces computing cost and increases efficiency by processing full photos in a single pass, in contrast to earlier object detection techniques that needed region proposals [11].

   YOLO-based object detection models are built on Convolutional Neural Networks (CNNs), which enable the network to learn intricate hierarchical characteristics for precise object recognition. YOLO's CNN integration guarantees strong detection performance under a range of image situations [14].
3. For people with vision impairments, depth prediction and object detection can greatly enhance navigation support. The study showed how accurate spatial awareness is made possible by combining object identification and depth sensing, enabling users to navigate securely [17].Because YOLO is a fast and accurate object detector, it is ideal for computer vision applications. We connect YOLO to a webcam and ensure that it keeps working in real time. [1]

YOLO Architecture

The YOLO architecture consists of An object detection-specific convolutional neural network (CNN) by making predictions directly from full images. Over its iterations (YOLOv1, YOLOv2, YOLOv3, and beyond), YOLO has evolved to become faster and more accurate while retaining its core architecture.

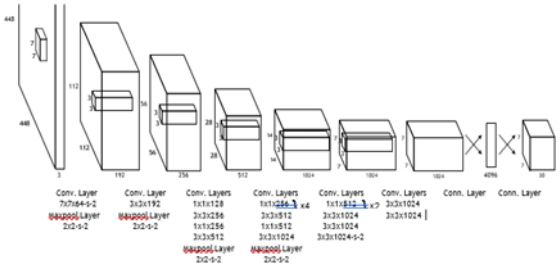

Fig 1 shows Layer Architecture

1. Network Structure and Grid Division

YOLO divides the input image into a S×SS×S grid. Each grid cell is responsible for predicting items whose centres are inside each grid cell. YOLO's grid-based approach allows it to finish both localisation and classification in a single forward pass. YOLO's development has tackled a number of issues, including increasing detection precision, decreasing false positives, and maximising computing effectiveness. With architectural improvements in every generation, YOLO is a popular option for a variety of real-world uses, including as autonomous navigation, medical imaging, and surveillance [4].

2. Bounding Box Prediction

Each cell predicts a fixed number of bounding boxes (typically, 2 to 5). Each bounding box prediction consists of:

Coordinates (x,y,w,h*x*,*y*,*w*,*h*): Relative to the cell's location, where x*x* and y*y* denote the box center, and w*w* and h*h* denote the w into h of the box.

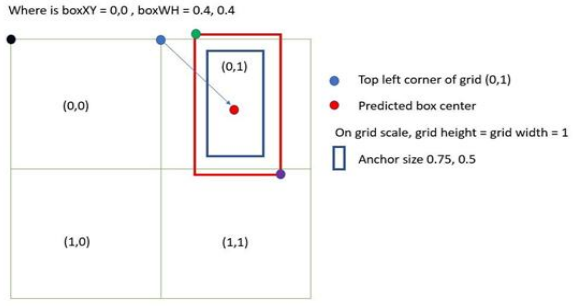

Fig 2 shows Bounding Box Recognition

Raspberry Pi-based machine learning-based object detection offers an effective and affordable edge computing solution. The study showed how real-time

object detection may be used with smart devices to automate robotics, assistive technologies, and surveillance procedures [7].

A Raspberry Pi and YOLO-based blind navigation support system was created to increase the mobility of people with visual impairments. The system ensures safe and autonomous navigation in unknown situations by detecting impediments and giving the user real-time feedback[8].

3. Class Prediction

YOLO predicts class probabilities for each cell. Since each cell only predicts one set of class probabilities, YOLO is designed for situations where one object dominates each cell, which can be a limitation when multiple small objects are clustered together.

4.Loss Function

YOLO's loss function combines three components:

Localization Loss: Penalizes errors in predicted bounding the box coordinates.

- Confidence Loss: Penalizes differences between predicted and actual confidence scores.
- Classification Loss: Penalizes errors in class predictions.

5. Anchor Boxes and Multi-Scale Detection

In later versions, YOLO introduces anchor boxes, which are predefined bounding boxes that the network adjusts for various objects. Additionally, YOLOv3 and newer versions use multi-scale predictions, enabling detection of objects at different resolutions to handle large and small objects more effectively.

6. Improvements in Later Versions

YOLO's groundbreaking architecture and single-pass detection approach offer a balance between speed and accuracy, making It is very adaptable to a wide range of applications that need real-time object identification. through iterative iterations. It is one of the most extensively used object detection models because to its cohesive strategy, efficacy, and efficiency.

YOLOv2 introduces improvements such as batch normalization, higher resolution input, and anchor boxes for better detection.

YOLOv3 and YOLOv4 incorporate features like Darknet-53 and CSP Darknet, respectively, as backbone networks. These versions introduce feature pyramid networks (FPN) for better multi-scale detection.

YOLO's efficacy in real-time road safety applications was demonstrated when it was integrated into autonomous cars for pothole and obstacle identification. In order to enhance vehicle navigation and passenger safety, the system effectively detected risks and sent out timely alerts [24].

By recognising and identifying items on shelves, a YOLO-based smart supermarket assistance system was created to assist people with visual impairments in navigating retail settings. By offering real-time object recognition feedback, our technology increased the freedom and accessibility of shopping for those with visual impairments [25].

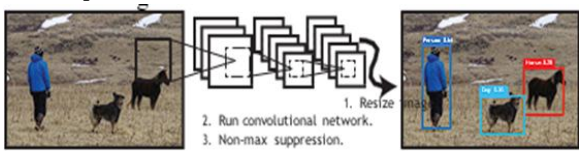

Fig 3 shows Bounding Box Recognition

In YOLO (You Only Look Once), Instead of being described as a classification or segmentation task, object detection is framed as a single regression problem. It is a well-liked option for real-time applications as it integrates object location and classification into a single neural network, optimizing for both speed and accuracy. This is a thorough explanation. of YOLO's working and methodologies.

Image Grid and Bounding Prediction

The input picture is divided into a S×SS×S grid by YOLO. If an object's center falls inside a grid cell, that cell is in charge of detecting it. Predicting the object's class and the bounding box coordinates for every grid cell is the primary goal.

For each grid cell, YOLO predicts:

Coordinates of the Bounding Box (x,y,w,hx,y,w,h): where ww and hh stand for the bounding box's width and height, usually expressed as ratios of the total picture dimensions, and xx and yy are the coordinates of the bounding box's center (with respect to the grid cell).

Class Probabilities: A set of class probabilities, or the likelihood of each class given the presence of an item, are predicted by each grid cell.Each grid cell's final prediction is a vector that contains the class probabilities, confidence score, and bounding box predictions. A few post-processing procedures are then applied to this vector to provide final object detections.

Anchor Boxes

Predefined boundary boxes are known as anchor boxes. with fixed aspect ratios, designed to help the model detect objects of various shapes and sizes. Instead of predicting arbitrary box dimensions, YOLO adjusts anchor boxes to fit objects better. In versions from YOLOv2 onward, the model leverages these anchor boxes:

A comparative study between YOLO and conventional CNN-based object detection models revealed that YOLO provides superior speed and efficiency, making it a better choice for real-time applications such as autonomous vehicles, security surveillance, and assistive technology for the visually impaired [16].

YOLO was found to be faster and more efficient than conventional CNN-based object identification models, making it a superior choice for real-time applications such as driverless automobiles, security surveillance, and assistive technology for the blind [16].Multiple Anchor Boxes per Grid Cell: Each grid cell has several anchor boxes, and each anchor is in charge of identifying diverse items of different sizes and shapes. For example, a tall anchor box may be better suitable for identifying a standing person, but a wide anchor box may be better suited to detecting vehicles.

PP-YOLO, an optimised version of YOLO, added various enhancements, including improved anchor-free mechanisms and optimised network designs. These changes produced a more efficient detection pipeline suitable for embedded and low-power devices [19].

MobileNetV2's lightweight architecture allows efficient deep learning models to run on edge devices like the Raspberry Pi. MobileNetV2 achieves good performance while requiring minimum processing resources [20].

Loss Function

YOLO’s function is a combination of different loss components to optimize localization, confidence, and classification.

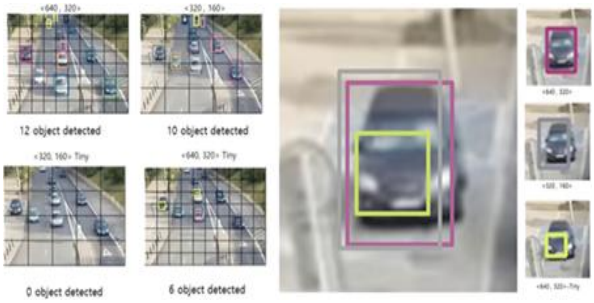

Fig 4 shows Grid Box Prediction

Localisation Loss: This is the error in anticipated bounding box coordinates (centre location x,yx,y, width ww, and height hh). It is commonly calculated as the Mean Squared Error (MSE) between the actual and predicted values, punishing incorrect bounding box sizes and placements.

Confidence Loss: There are two components to the confidence loss: one for cells that contain objects and one for those that do not. It penalises inaccurate objectness predictions, with the confidence score reflecting both the likelihood of the observed object and the correctness of the bounding box.

Classification Loss: This metric quantifies the difference between the expected and actual class probabilities for each object. Cross-entropy loss is a frequent method for penalising inaccurate class predictions.

The end product is an intriguing and interactive system. Although YOLO analyzes each image separately, when connected to a camera, it acts as a tracking device, identifying objects as they move and change in appearance. Our project website, http://pjreddie.com/yolo/, has a system demo and the source code.[1]

YOLO’s loss function is defined to balance these three components, ensuring that the network optimizes for both accurate localization and correct classification.

Non-Maximum Suppression (NMS)

To reduce duplicate detections and false positives, YOLO uses Non-Maximum Suppression (NMS):

Suppression of Overlapping Boxes: Anchor boxes cause YOLO to generate several forecasts for the same item. NMS assists by hiding all of an object's bounding boxes save the one with the highest confidence score.

NMS plays crucial role in filtering redundant boxes and ensuring only the most confident detections are kept.

Thresholding: NMS applies a threshold to the Union over Intersection metric. If two boxes have an IoU larger than a certain threshold, the box with the lowest confidence score is removed. Thresholding: NMS applies a threshold to the Union over Intersection metric. If two boxes have an IoU larger than a certain threshold, the box with the lowest confidence score is removed.

TABLE I. Comparison between Yolo and Yolo_v3

S.NO	CRITERIA	YOLO	YOLO_V3
1	No of Layers	Yolo has 24 convolutional layers followed by 2 fully connected layers	Yolo_v3 has 106 convolutional layers (53 original layers and 53 detection layers are added)
2	Shape of Detection Kernel	(S, S, B x 5+C)	1 x 1 x (B x (5+C))
3	No. of Predictions	Yolo gives only one prediction at the last layer by applying 1x1 detection kernel on a feature map.	Yolo_v3 gives predictions at three scales by applying 1 x 1 detection kernel at three different positions and sizes.
4	Detection of Smaller Objects	Yolo is not good in detecting small objects	Yolo_v3 preserve the features which helps in detecting small objects from the complex images.
5	Choice of Anchor Boxes	No. of anchor boxes is not fixed it makes use of K-means algorithm to generate anchors	Yolo_v3 make use of 9 anchor boxes i.e. 3 for each scale
6	Softmaxing	Yolo make use of softmax to select the object with maximum confidence score.	Yolo_v3 replaces softmax concept by performing multi label classification of objects.
7	Classification loss	It uses mean square error (SSE) for calculating classification loss.	It uses binary cross entropy for calculating loss.
8	Class Prediction	It uses Softmax function to calculate probability of object falling under particular category by converting scores that sum up to one.	Class Predictions and Object Confidence are Predicted through Logistic Regression.
9	Mean average Precision	23.7 %	55.3 %
10	Speed	Low	Medium
11	Accuracy	Medium	High

Feature Extraction Backbone (Darknet)
YOLO models use different versions of Darknet as the backbone network:

Darknet-19 (YOLOv2): balances speed and accuracy with its 19 convolutional layers and 5 max-pooling layers.

Darknet-53 (YOLOv3): A deeper architecture with 53 convolutional layers, incorporating residual blocks for improved gradient flow and accuracy.

CSPDarknet-53 (YOLOv4): Further refined with Cross Stage Partial connections to reduce memory cost and increase computation efficiency, enabling YOLOv4 to perform well on resource-constrained devices.

The backbone network in YOLO extracts feature maps from the input picture, which are subsequently utilized by further layers for bounding box prediction.

Model	Size	mAP^{val} 0.5:0.95	$Speed^{T4}$ trt fp16 b1 (fps)	$Speed^{T4}$ trt fp16 b32 (fps)	Params (M)	FLOPs (G)
YOLOv6-N	640	37.5	779	1187	4.7	11.4
YOLOv6-S	640	45.0	339	484	18.5	45.3
YOLOv6-M	640	50.0	175	226	34.9	85.8
YOLOv6-L	640	52.8	98	116	59.6	150.7

Fig 5 shows Yolo V evolution

Multi-Scale Detection (Feature Pyramid Networks)
YOLOv3 and later versions introduced multi-scale detection to increase the precision of detection, particularly for objects of different sizes:

Multiple Feature Maps: YOLO leverages feature maps at multiple resolutions, typically from three layers, to detect small, medium, and large objects. This is achieved by predicting bounding boxes at different scales using feature maps from different stages of the network.

Feature Pyramid Networks (FPN): By concatenating features from several layers, FPNs enable YOLO to recognize objects at different sizes, guaranteeing that semantic information (from deeper layers) is integrated with smaller details (from earlier levels).

Improved Training Techniques
To enhance the training process, YOLO incorporates techniques such as:
Batch Normalization: YOLO uses batch normalization to reduce internal covariate shifts and make training more stable.

Data Augmentation: Various forms of augmentation, including random scaling, cropping, and flipping, are used to improve the model's robustness.

Class Imbalance Handling: YOLO addresses class imbalances through techniques like focal loss (YOLOv4), which emphasizes difficult-to-classify samples more heavily.

Incremental Improvements in YOLO Versions
Each successive version of YOLO introduces architectural and training enhancements to improve the accuracy and speed:

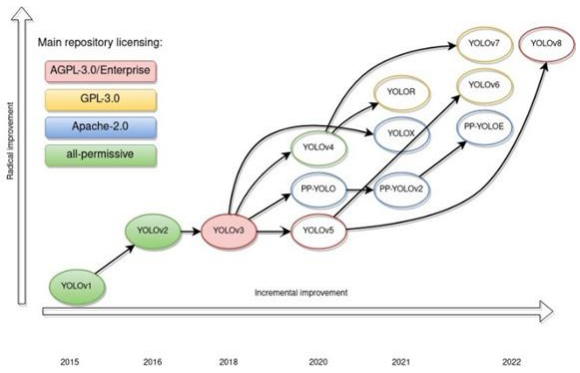

Fig 6 shows Test Analysis of YoloV

YOLOv2 (YOLO9000): Incorporates anchor boxes, high-resolution images, and multi-class object detection (up to 9000 classes).

YOLOv3: Adds multi-scale detection, a deeper backbone (Darknet-53), and For each feature map, three anchor boxes are used.

YOLOv4: Optimizes the training with Cross Stage Partial (CSP) networks, a robust activation function (Mish), and Self-Adversarial Training (SAT) for data augmentation.

YOLOv5: Introduces additional optimizations for speed and accuracy, further refining the detection head and training pipeline.

YOLOv7: Improves the architectural layout, providing even faster speeds and better accuracy by focusing on efficient model scaling and parameter optimization.

Benefits of Yolo

Real-Time Detection:

YOLO is extremely fast, capable of real-time detection at up to 45 frames per second (fps) in its fastest versions. This is highly beneficial for applications that require instantaneous response times, such as autonomous driving, robotics, and video surveillance.

Single-Pass Detection:

Unlike region-based methods like R-CNN that perform multiple passes through an image (region proposal, classification, and refinement), The picture is processed by YOLO in a single forward pass. YOLO is appropriate for applications with constrained computer resources because of its unified approach, which drastically lowers computational costs.

Simplified Architecture:

YOLO is a single, end-to-end neural network. The simplicity of its design makes it relatively easy to implement, deploy, and fine-tune for various applications, and it avoids the requirement for intricate pipelines that are employed in other object identification techniques.

Good Generalization:

YOLO's design encourages it to learn generalized representations of objects, making it capable of detecting objects in unusual orientations or partially occluded states. This property makes YOLO more robust in diverse environments.

Wide Range of Applications:

Due to its high speed and excellent harmony between computational efficiency and accuracy, YOLO is widely adopted across applications like video processing, drone navigation, human-computer interaction, retail automation, and assistive technologies.

Limitations of the YOLO model

Lower Accuracy on Small Objects:

YOLO finds it difficult to detect small objects, especially when multiple small objects are clustered together. The grid-based structure, where each grid cell is responsible for detecting only one object, limits YOLO's capacity to handle dense object clusters.

Coarse Localization:

YOLO has limitations in achieving fine-grained localization for objects. This can result in slightly imprecise bounding boxes, especially in complex images with many overlapping objects.

Class Imbalance Issues:

YOLO might struggle to deal with class imbalances, especially when certain items (such as humans) appear far more frequently than others. YOLO's detection accuracy can still be affected by imbalanced datasets, despite improvements in subsequent versions.[1]

REFERENCES

[1] Hussain, M. "YOLOv5, YOLOv8 and YOLOv10: The Go-To Detectors for Real-time Vision." *arXiv*

preprint arXiv:2407.02988, 2024.

[2] Amang, S., Sen, B., Pradhan, A., Sharma, K., & Singh, V. K. "Enhancing COVID-19 Safety: Exploring YOLOv8 Object Detection for Accurate Face Mask Classification." *International Journal of Intelligent Systems and Applications in Engineering*, vol. 11, no. 2, pp. 892–897, 2023.

[3] Ng, G. J., Goil, A. K., Chan, H., Lee, X. C., Mustaffa, R. B., Jason, T., Woon, Z. T., & Shen, B. "A Novel Application for Real-Time Arrhythmia Detection Using YOLOv8." *arXiv preprint arXiv:2305.16727v2*, 2023.

[4] Diwan, T., Anirudh, G., & Tembhurne, J. V. "Object detection using YOLO: Challenges, architectural successors, datasets and applications." *Multimedia Tools and Applications*, vol. 82, no. 6, pp. 9243–9275, 2023.

[5] Sirisha, U., Praveen, S. P., Srinivasu, P. N., et al. "Statistical Analysis of Design Aspects of Various YOLO-Based Deep Learning Models for Object Detection." *International Journal of Computational Intelligence Systems*, vol. 16, p. 126, 2023.

[6] Aiswal, T., Pandey, M., & Tripathi, P. "Real-Time Multiple Object Detection Using Raspberry Pi and Tiny-ML Approach." *Recent Advances in Electrical & Electronic Engineering*, 2024.

[7] Li, Z., Zhang, Y., Zhang, J., Liu, F., & Chen, W. "A Multi-Sensory Guidance System for the Visually Impaired Using YOLO and ORB-SLAM." *Information*, vol. 13, no. 7, p. 343, 2022.

[8] Ahmad, T., et al. "Object detection through modified YOLO neural network." *Scientific Programming*, vol. 2020, Article ID 8403262, 2020.

[9] Lee, Y.-H., & Kim, Y. "Comparison YOLO with CNN for Object Detection." *Journal of the Semiconductor & Display Technology*, vol. 19, no. 1, pp. 85–92, 2020.

[10] Long, X., Deng, K., Wang, G., Zhang, Y., Dang, Q., Gao, Y., Shen, H., Ren, J., Han, S., Ding, E., & Wen, S. "PP-YOLO: An Effective and Efficient Implementation of Object Detector." *arXiv preprint arXiv:2007.12099*, 2020.

[11] Zhao, Z., Zheng, P., Xu, S., & Wu, X. "Object Detection with Deep Learning: A Review." *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[12] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. "SSD: Single Shot MultiBox Detector." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 4, pp. 761–774, 2016.

[13] Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollar, P. "Focal Loss for Dense Object Detection." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020.

[14] Tan, M., Pang, R., & Le, Q. V. "EfficientDet: Scalable and Efficient Object Detection." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 11, pp. 4013–4021, 2021.

[15] Ren, S., He, K., Girshick, R., & Sun, J. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[16] Zhou, X., Wang, D., & Krähenbühl, P. "Objects as Points." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 8, pp. 2560–2572, 2021.

[17] Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. "Deep Learning for Generic Object Detection: A Survey." *International Journal of Computer Vision*, vol. 128, no. 2, pp. 261–318, 2020.

[18] Zou, Z., Shi, Z., Guo, Y., & Ye, J. "Object Detection in 20 Years: A Survey." *International Journal of Computer Vision*, vol. 129, no. 1, pp. 161–252, 2021.

[19] Cai, Z., & Vasconcelos, N. "Cascade R-CNN: Delving into High Quality Object Detection." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 5, pp. 1483–1498, 2021.

[20] **Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A ., & Zagoruyko, S.** "End-to-End Object Detection with Transformers." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 3, pp. 798–812, 2022.

[21] Redmon, J., & Farhadi, A. "YOLOv3: An Incremental Improvement." *arXiv preprint arXiv:1804.02767*, 2018.

[22] Bochkovskiy, A., Wang, C. Y., & Liao, H. M. "YOLOv4: Optimal Speed and Accuracy of Object Detection." *arXiv preprint arXiv:2004.10934*, 2020.

[23] Liu, C., et al. "Object Detection Based on YOLO Network." *IEEE Transactions on Circuits and*

Systems for Video Technology, vol. 29, no. 8, pp. 4054–4061, 2019.

[24] Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. "MobileNetV2: Inverted Residuals and Linear Bottlenecks." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 3, pp. 651–665, 2022.

[25] Edmon, J., Divvala, S., Girshick, R., & Farhadi, A. "You Only Look Once: Unified, Real-Time Object Detection." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 779–785, 2017.