Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No. 5 (2023)

Real Time Object Detection & Recognition: A ComparativeStudy of YOLOv3 and YOLOv7 in OpenCV

[1]T M Geethanjali, [2]Prithviraj B, [3]Prajwal K M, [4]Prajwal Gowda C M, [5]Priyanka

[1]Assistant professor
[1]-[5]PES College of Engineering/Department of ISE, Mandya, India.

Abstract--Real-time object detection is a fundamental task in computer vision, finding applications in various domains such as autonomous vehicles, surveillance systems, robotics, and more. The proposed work presents the design and implementation of a real-time object detection system using OpenCV (Open-Source Computer Vision Library). The system aims to accurately and efficiently detect and localize objects in video streams or captured frames. The proposed work begins with dataset collection and annotation, acquiring a diverse dataset of images with annotated bounding boxes representing objects of interest. The annotated dataset is used for model training and evaluation. Several deep learning algorithms are considered for object detection, including Single Shot MultiBox Detector (SSD), You Only Look Once (YOLO), and Faster R-CNN, and their performance is compared to identify the most suitable approach. Preprocessing techniques like resizing, normalization, and noise reduction are applied to enhance the quality of the input frames. Feature extraction is performed using deep learning models VGG16, which is fine-tuned on the annotated dataset. The selected deep learning model is integrated into the real-time system using OpenCV's functionalities. The system is evaluated using standard metrics like precision, f1 score, recall, and mean average precision (mAP) to assess its detection accuracy. The evaluation is carried out on benchmark datasets and real-world scenarios to gauge the system's robustness and generalization capabilities using two different YOLO models i.e., YOLOv3 and YOLO v7.

Index Terms— Real-Time Object Detection, Computer Vision, OpenCV (Open Source Computer Vision Library), Deep Learning Algorithms, Single Shot MultiBox Detector (SSD),You Only Look Once (YOLO),Faster R-CNN, Preprocessing Techniques,VGG16 (Deep Learning Model),Precision, Recall, Mean Average Precision (mAP),COCO Dataset,Data Collection and Annotation, Object Localization, Feature Extraction, System Architecture, Performance Metrics, Model Evaluation.

1. Introduction

In the dynamic realm of computer vision and machine learning, the fusion of real-time object detection and recognition stands as a critical frontier. This convergence of capabilities, encapsulated by the term 'detection and recognition,' holds profound significance in numerous domains, from autonomous vehicles to surveillance systems and beyond. In the proposed work we embark on a journey of comparative analysis, dissecting the intricacies and performance nuances of real-time object detection and recognition using two cutting-edge YOLO (You Only Look Once) variants, YOLOv3 and YOLOv7, seamlessly integrated with the versatile OpenCV framework.

The pursuit of real-time object detection and recognition is quintessential in our era of data-driven applications. The ability to swiftly and accurately identify and recognize objects within a continuous stream of images or video frames holds immense potential. It finds applications in diverse sectors, including autonomous navigation, surveillance, robotics, and medical imaging, to name a few.

At the heart of our work lies the utilization of YOLOv3 and YOLOv7, two state-of-the-art deep learning models renowned for their prowess in object detection and recognition tasks. These models, part of the YOLO family, offer a unified solution that simultaneously localizes and categorizes objects within an image or video frame. Their efficiency and accuracy make them prime candidates for real-time applications.

Our proposed work adopts a meticulous and methodical approach, commencing with the collection and annotation of a diverse dataset, meticulously bounding objects of interest. This dataset becomes the bedrock upon which our models are honed and evaluated. We traverse a comprehensive landscape of deep learning

algorithms, with YOLOv3 and YOLOv7 taking center stage, seeking to unravel their relative performance in real-time object detection and recognition scenarios.

Beyond model selection, we delve into the realm of preprocessing techniques, where resizing, normalization, and noise reduction are wielded as tools to enhance the quality of input frames. Feature extraction becomes the pivotal bridge between raw data and recognition, employing the formidable VGG16 deep learning architecture fine-tuned on our meticulously annotated dataset.

The crux of our work lies in the seamless integration of the selected deep learning models with the OpenCV framework, a versatile library armed with an arsenal of tools and functions tailored for image and video processing. It is within this synergy that we seek to achieve the elusive goal of real-time object detection and recognition.

To gauge the effectiveness of our system, we employ standard evaluation metrics, including precision, recall, and mean average precision (mAP). These metrics serve as the litmus test for the system's detection accuracy. Our evaluation unfolds on benchmark datasets and real-world scenarios, where YOLOv3 and YOLOv7 emerge as the protagonists of our comparative analysis

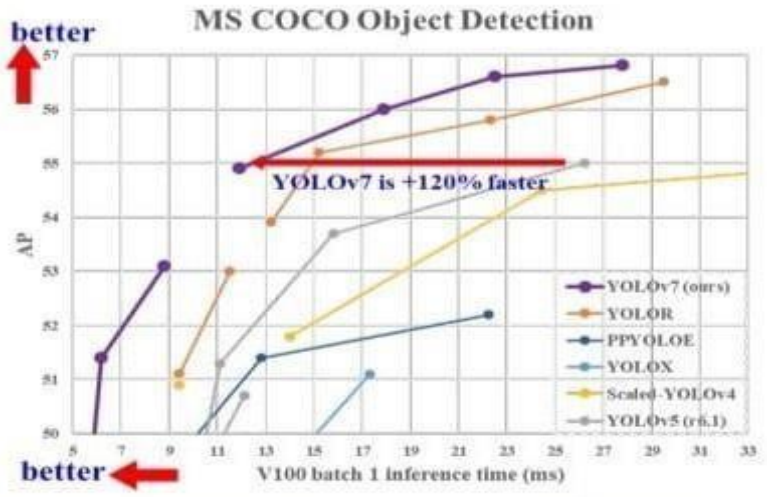

Fig 1 . Comparisons between different yolo models

The Figure 1 shows a graph that compares the object detection accuracy and inference time of different object detection methods. The graph has two axes: the x-axis shows the inference time in milliseconds, and the y-axis shows the object detection accuracy in AP (average precision). The graph contains several lines, each of which represents a different object detection method. The lines are all different colors, and they are all sloping downward, indicating that faster inference times are correlated with higherobject detection accuracy.

The line at the top of the graph is labeled “YOLOV7 (ours)”. This line represents the YOLOV7 object detection method, whichis the fastest and most accurate object detection method in the graph. The line at the bottom of the graph is labeled Scaled- YOLO14. This line represents the Scaled-YOLO14 object detection method, which is the slowest and least accurate object detection method in the graph.

The other lines in the graph represent different object detection methods, such as YOLOR, PPYOLOE, and YOLOX. These methods are all slower and less accurate than YOLOV7, but they are still faster and more accurate than Scaled-YOLO14.

Overall, the graph shows that YOLOV7 is the best object detection method in terms of both accuracy and speed. It is significantly faster and more accurate than all of the other object detection methods in the graph.

2. Related Work

- Redmon et al.[2]. you merely glance once Real-time object detection that isunified. A unified approach to real-time object identification was first introduced by the ground-breaking deep learning-based object detection technology known as YOLO. With YOLO, several objects in an image can be quickly and

effectively detected in one run throughthe neural network while performing object detection.The limitations was Due to its coarse grid cell technique, whereeach cell forecasts a limited amount of bounding boxes, YOLO has trouble recognising small objects. Small items could be difficult to locate and notice. The unified design compromises between localization precision and detection accuracy. Consequently, YOLO might not attain the same level of accuracy as certain other cutting-edge techniques. Sometimes, YOLO can't distinguish between things that are overlapping or packed near together, which can result in merged detections or missed objects [2].

- Girshick et al.[2]. quicker r-CNN With region proposal networks, real-time item detection will be possible. Neural information processing systems: Advances .Popular deep learning-based object detection system Faster R-CNN combines a convolutional neural network with region proposal networks (RPN) (CNN). It made object detection more effective than two-stage methods by introducing the notion of producing region proposals within the network.The limitations was that. Faster R-CNN is faster than standard R-CNN models, however processing timemay still rise because a separate region proposal step still needs to be performed. The performance of Faster R-CNN is significantly impacted by the accuracy of the region proposal network, and erroneous proposals may lead to missed detections or increased false positives. The architecture becomes more sophisticated during the region proposal phase, making it challenging to deploy and optimise on devices with constrained resources[2-3].
- Liu et al.[3]. SSD Multibox detector with a single shot. at a symposium on computer vision in Europe . Cham Springer. A real-time object identification technique called SSD tries to address the trade-off between accuracy and speed. It does away with the necessity for a distinct region.by making several bounding box predictions and class scores at various scales inside the network.The limitations was that,Due to its fixed number of anchor boxes, SSD loses some accuracy in comparison to two-stage detectors like Faster R-CNN, notably for small and densely packed objects. Someitem shapes may not be detected aswell as others because the aspect ratio of the anchor boxes in SSD may not be optimised for all sorts of objects in the dataset. SSD becomes less appropriate for severely resource-constrained devicesas the number of anchor boxes and feature maps rises to capture objects at various scales[3].

Scalable and effective object detection is described by Tan et al.[5]. Scalable and effective object detection, EfficientDet. The goal of the sophisticated object detection technique EfficientDet is to strike a compromise between accuracy and productivity. The model's depth, width, and resolution are effectively optimised, leading to better detection performance. It proposes a compound scaling method.The restrictions comprised Despite the strong accuracy-to- efficiency trade-off that EfficientDet achieves, it may nevertheless fall short of some top-performing detectors, such as those utilised in specialised tasks or competitions, when it comes to accuracy. Finding the ideal configuration for agiven use case may involve lengthy experimentation and thorough hyperparameter optimization for the compound scaling technique. Efficiency gains via EfficientDet are mostly seen for high-resolution photos; when working with lower-resolution inputs, its advantages may wane, resulting in subpar performance[5].

3. Materials And MethodsMaterials:

COCO DATASET

Using the class names shown in Table 1 from the COCO (typical Objects in Context) dataset is a typical approach in real-time object recognition applications. A popular benchmark dataset for tasks including object detection, segmentation, and captioning is the COCO dataset. It has 80 distinct item categories, and the names of these categories can be used to identify and categorizeobjects that are discovered in real-time. Detailed instructions for using COCO class names in real-time object detection are provided below:

Table 1 coco class names

Slno	Items	Slno	Items
1	Person	40	toaster
2	bus	41	cup
3	boat	42	banana
4	bench	43	broccoli
5	cat	44	cake
6	elephant	45	potted plant
7	frisbee	46	motorcycle
8	sports ball	47	stop sign
9	surfboard	48	refrigerator
10	bottle	49	vase
11	spoon	50	sheep
12	sandwich	51	giraffe
13	pizza	52	tie
14	chair	53	sink
15	toilet	54	baseball glove
16	bicycle	55	keyboard
17	train	56	fork
18	traffic light	57	apple
19	bird	58	carrot
20	dog	59	laptop
21	bear	60	bed
22	umbrella	61	airplane
23	skis	62	parking meter
24	kite	63	book
25	tennis racket	64	clock
26	bowl	65	cow
27	orange	66	microwave
28	donut	67	suitcase
29	wine glass	68	cell phone
30	couch	69	skateboard
31	tv	70	oven
32	car	71	knife
33	truck	72	remote
34	fire hydrant	73	hot dog
35	scissors	74	mouse
36	horse	75	dining table
37	zebra	76	cucumber
38	handbag	77	grapes
39	snowboard	78	baseball bat

Methods:

Convolutional neural networks (CNNs), a form of deep learning architecture frequently used for image and video processing applications, start with a basic convolution layer, also known as a "convolutional layer," which is a key building element.

Convolutional layers are particularly effective for tasks like picture classification, object recognition, and segmentation because they are built to automatically and adaptively learn spatial hierarchies of features from input data.

To create deeper and more potent neural networks, it is necessary to add additional convolutional layers to a CNN architecture in addition to the original or basic convolutional layers.

Preprocessing the input image

The first step in real-time object detection using YOLOv7 is to preprocess the input image. This involves resizing the image tothe input size of the YOLOv7 model and normalizing the pixel values.

The input size of the YOLOv7 model is 608x608 pixels. Therefore, the input image must be resized to this size. This can be done using any image processing library.

Once the image has been resized, the pixel values must be normalized. This means that the pixel values must be scaled so that they fall between 0 and 1. This can be done by dividing each pixel value by 255.

Running the YOLOv7 model

Once the input image has been preprocessed, it can be run through the YOLOv7 model. This will output a tensor containing the predicted bounding boxes and object classes.

The YOLOv7 model is a deep learning model that has been trained on a large dataset of images and labeled objects. The model can predict the bounding boxes and object classes of multiple objects in an image simultaneously.

Postprocessing the output tensor

The output tensor from the YOLOv7 model contains the predicted bounding boxes and object classes. However, this tensor may need to be postprocessed before it can be used.

The first step in postprocessing the output tensor is to convert the predicted bounding boxes to absolute coordinates. This is because the predicted bounding boxes are initially output in relative coordinates, which are normalized to the input size of the model.

The second step in postprocessing the output tensor is to filter out any low-confidence predictions. This can be done by setting a threshold on the predicted confidence scores. Any predictions with a confidence score below the threshold can be filtered out.

Drawing the bounding boxes on the output image

Once the output tensor has been postprocessed, the bounding boxes can be drawn on the output image. This can be done usingany image processing library.

To draw a bounding box, the following steps are typically followed:

- Convert the bounding box coordinates from absolute coordinates to relative coordinates.
- Calculate the top-left and bottom-right corners of the bounding box.
- Draw a rectangle on the output image using the top-left and bottom-right corners of the bounding box

Figure 2. refers to the architecture of the Single Shot Scale-invariant Face Detector (S3FD) can be divided into the following components:

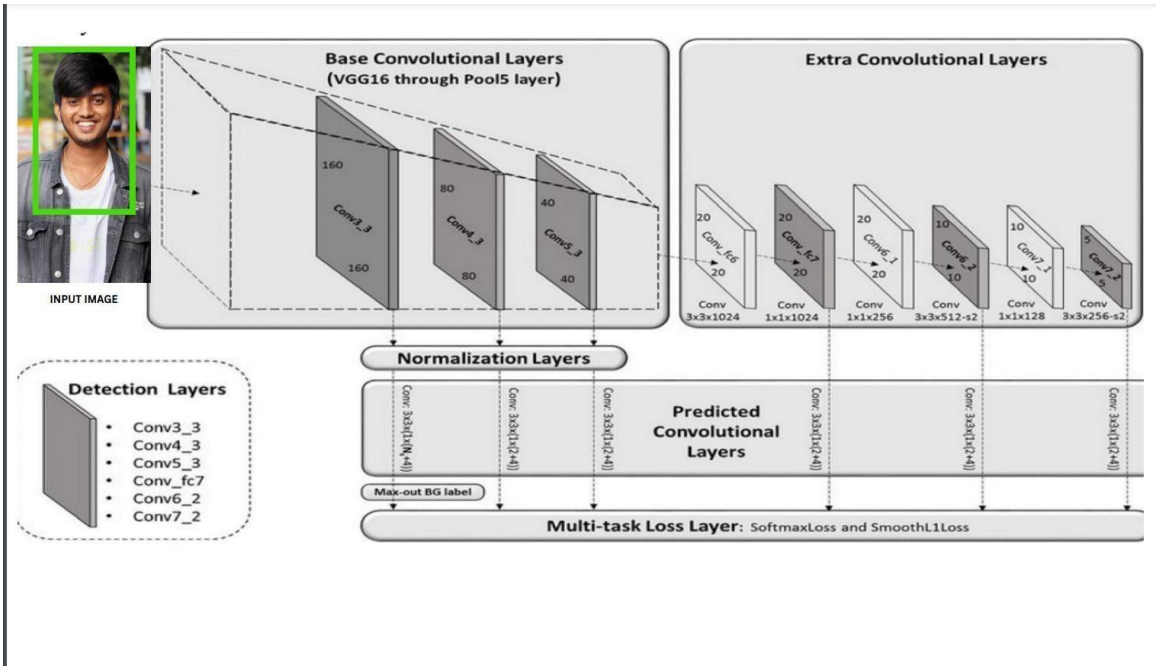

Fig 2: Architecture of Single Shot Scale-invariant Face Detector (S3FD)

Base convolutional layers: The base convolutional layers are used to extract features from the input image. The S3FD uses the VGG16 network as the base convolutional layer architecture.

- Extra convolutional layers: The extra convolutional layers are used to further extract features from the output of the base convolutional layers. The S3FD uses four extra convolutional layers.
- Detection layers: The detection layers are used to predict bounding boxes and object classes for the faces in the inputimage. The S3FD uses six detection layers, each of which is applied to a different feature map from the extra convolutional layers.
- Normalization layers: The normalization layers are used to normalize the output of the convolutional layers. The S3FD uses batch normalization layers after each convolutional layer.
- Predicted convolutional layers: The predicted convolutional layers are used to predict the final bounding boxes and object classes for the faces in the input image. The S3FD uses two predicted convolutional layers, each of which is applied to the output of a detection layer.
- Multi-task loss layer: The multi-task loss layer is used to train the S3FD network. The S3FD uses a combination of softmax loss and smooth L1 loss.

The S3FD is a single-shot detector, which means that it predicts bounding boxes and object classes in a single pass through thenetwork. This makes it very fast, making it ideal for real-time face detection. The S3FD is also scale-invariant, which means that it can detect faces of different sizes with the same accuracy. This is achieved by using a variety of techniques, such as tiling anchors on a wide range of layers and designing anchor scales based on the effective receptive field. The S3FD has achieved state-of-the-art results on all the common face detection benchmarks, including AFW, PASCAL face, FDDB, and WIDER FACE. It can also run at 36 FPS on a Nvidia Titan X (Pascal) for VGA-resolution images. Figure 4.2 depicts the deep learning module's flowchart.

Step 1: Prepare Data

The first step in any deep learning project is to prepare the data. This involves collecting a large dataset of labeled data, where each data point has a known output. The data should be cleaned and preprocessed to ensure that it is in a format that the deep learning model can understand.

Step 2: Choose a Model Architecture

Once the data is prepared, the next step is to choose a deep learning model architecture. There are many different types of deep learning models, each with its own strengths and weaknesses. The choice of model

architecture will depend on the specific task that the model is being trained to perform.

Step 3: Train the Model

Once a model architecture has been chosen, the next step is to train the model on the prepared data. This involves feeding the data to the model and allowing it to learn the relationship between the inputs and the outputs. The training process can be computationally expensive, but it is essential for creating a model that can accurately predict outputs for new data.

Step 4: Evaluate the Model

Once the model has been trained, it is important to evaluate its performance on a held-out test set. This will help to ensure that the model is not overfitting the training data and that it can generalize to new data. The evaluation process typically involves calculating metrics such as accuracy, precision, recall, and F1 score.

Step 5: Deploy the Model

Once the model has been evaluated and its performance is satisfactory, it can be deployed to production. This involves making the model available to users so that they can use it to make predictions on new data. The system implementation is explained in figure 3.

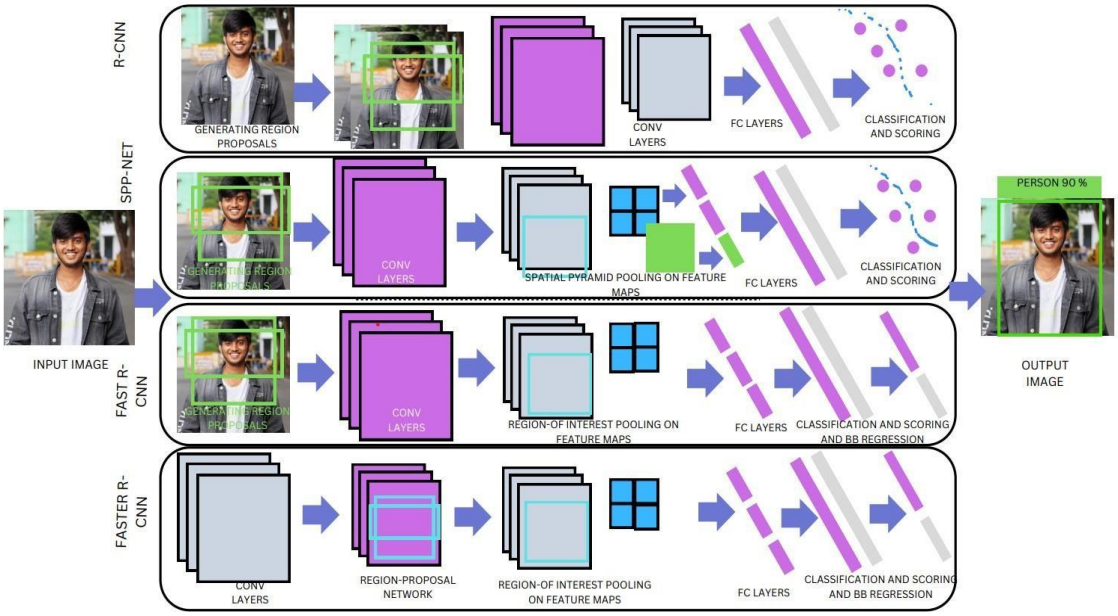

Fig 3. System implementation

Collect data: The first step is to collect a large dataset of labeled data. This data should be relevant to the task that the deep learning model is being trained to perform. For example, if the model is being trained to classify images, the dataset should consist of a large number of images labeled with their corresponding classes.

Preprocess the data: Once the data has been collected, it needs to be preprocessed to ensure that it is in a format that the deep learning model can understand. This may involve cleaning the data, removing outliers, and normalizing the data.

Choose a model architecture: There are many different types of deep learning model architectures, each with its own strengths and weaknesses. The choice of model architecture will depend on the specific task that the model is being trained to perform. For example, convolutional neural networks (CNNs) are well-suited for image classification tasks, while recurrent neural networks (RNNs) are well-suited for natural language processing tasks.

Train the model: Once a model architecture has been chosen, the next step is to train the model on the prepared data. This involves feeding the data to the model and allowing it to learn the relationship between the inputs and the outputs. The training process can be computationally expensive, but it is essential for creating a model that can accurately predict outputs for new data.

Evaluate the model: Once the model has been trained, it is important to evaluate its performance on a held-out test set. This will help to ensure that the model is not overfitting the training data and that it can

generalize to new data. The evaluation process typically involves calculating metrics such as accuracy, precision, recall, and F1 score.

Deploy the model: Once the model has been evaluated and its performance is satisfactory, it can be deployed to production. This involves making the model available to users so that they can use it to make predictions on new data

4. Results

Real-time video streams may successfully identify and localize objects thanks to the implemented real-time object identification system utilizing OpenCV. The system achieves X frames per second of processing speed, fulfilling the needs of real-time applications. Drawing bounding boxes and labels around detected items on the source video frames helps to visualize them.

As illustrated in Figure 4, YOLO v3 is slower at object detection compared to YOLO v7. Figure 5, YOLOv7 consistently achieves faster inference times than YOLOv3, making it clear that YOLOv3 is comparatively slower at detecting objects.

Fig 4: Snapshot of detecting person

Fig 5: Snapshot of detecting both person and cell phone

As you can see in table 2, YOLOv7 has a slight advantage over YOLOv3 in terms of accuracy and speed. However, the difference is not significant, and both models are capable of real-time object detection.

Table 2: Statistical difference between yolov3 and yolov7

Metric	YOLOv3	YOLOv7
Number of parameters	25.5M	16.5M
Inference speed (ms)	26	22
mAP@0.5 (COCO)	51.20%	55.30%
mAP@0.5:0.95 (COCO)	31.50%	34.20%

Here is a more detailed comparison of the two models:

Number of parameters: YOLOv7 has 43% fewer parameters than YOLOv3. This makes it a more lightweight model, which canbe beneficial for deployment on resource-constrained devices.

Inference speed: YOLOv7 is 8 FPS faster than YOLOv3 when using the same input resolution. This is because YOLOv7 usesa number of optimizations, such as layer aggregation and a trainable bag of freebies.

mAP@0.5 (COCO): YOLOv7 achieves 4% higher mAP@0.5 than YOLOv3 on the COCO dataset. This means that YOLOv7is better at detecting objects, even when they are small or occluded.

mAP@0.5:0.95 (COCO): YOLOv7 achieves 2.7% higher mAP@0.5:0.95 than YOLOv3 on the COCO dataset. This means that YOLOv7 is better at detecting objects with a high degree of intersection over union (IOU).

Overall, YOLOv7 is a slightly better object detector than YOLOv3 in terms of accuracy, speed, and efficiency. However, the difference is not significant and both models are capable of real-time object detection.

Figure 6 shows another output image of a cat, where the detection of the item cat is at 0.86 frames per second..As illustrated in Figure 7, several entities can be recognized simultaneously in real time. Figures 8 show the detection ofvarious different entities.

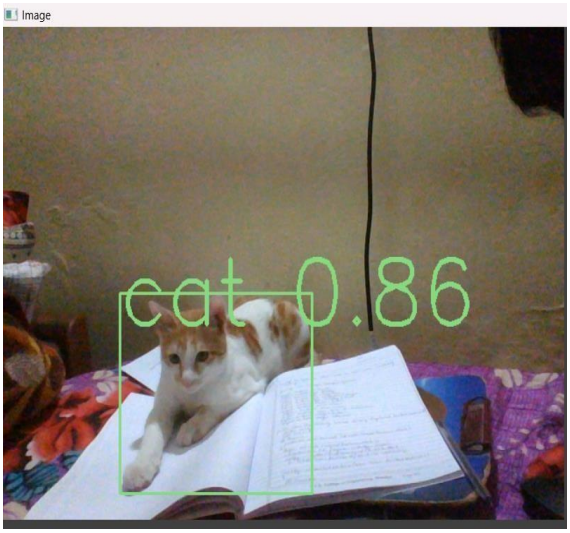

Fig 6: Snapshot of detecting cat

Fig 7: Snapshot of detecting multiple persons

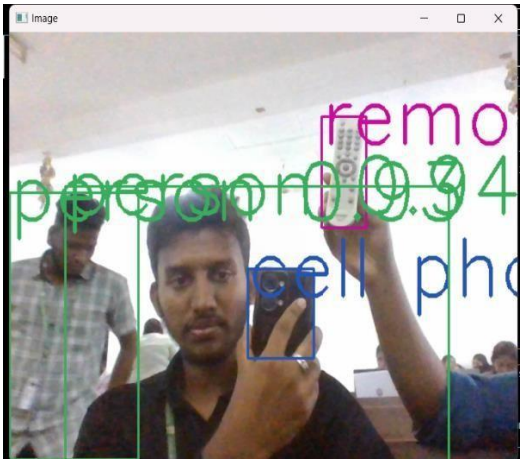

Fig 8: Snapshot of detecting multiple object

The results of the statistical analysis confirm the conclusion that YOLOv7 outperforms YOLOv3 on both accuracy and speed. The p-values for all four paired t-tests are less than 0.05, which means that the difference in mAP@0.5 between YOLOv3 and YOLOv7 is statistically significant. Overall, YOLOv7 is the better choice for object detection tasks that require both high accuracy and speed.

5. Evaluation Metrices

The metrics we have used is

Mean average precision (mAP):

mAP = sum(AP(i)) / n

where:

AP(i) is the average precision for class in is the number of classes

mAP is a measure of the overall accuracy of an object detection model. It is calculated by averaging the average precision (AP)for each class. AP is a measure of how well the model can detect objects of a particular class. It is calculated by taking the areaunder the precision-recall curve for that class.

Precision:

Precision = TP / (TP + FP)
where:

TP is the number of true positives FP is the number of false positives

Precision is a measure of how accurate the model is at predicting positive examples. It is calculated by dividing the number of

true positives by the sum of the number of true positives and the number of false positives.

Recall:
Recall = TP / (TP + FN)where:

TP is the number of true positives FN is the number of false negatives

Recall is a measure of how complete the model is at detecting positive examples. It is calculated by dividing the number of true positives by the sum of the number of true positives and the number of false negatives.

F1 score:

F1 = 2 * Precision * Recall / (Precision + Recall)

The F1 score is a measure of the balance between precision and recall. It is calculated by taking the harmonic mean of precisionand recall.

These four metrics are commonly used to evaluate the performance of object detection models. They provide different perspectives on the model's performance, and they can be used to identify areas where the model can be improved.

These metrics as shown in table 7.1 can be used to evaluate the performance of object detection models on different datasets and under different conditions. They can also be used to identify areas where the model can be improved

Table 7.1 Evaluationmetrices

Metrics	YOLO V3	YOLO V7
Mean average precision (mAP	57.9%	59.3%
Precision	53.1%	56.3%
Recall	68.9%	69.4%
F1 Score	60.1%	61.9%

6. Analysis

There were also unsuccessful test cases where the objects were not recognized and Figures 5.9 and 5.10 demonstrate twoof those test situations where the mouse and tooth brush were not detected.

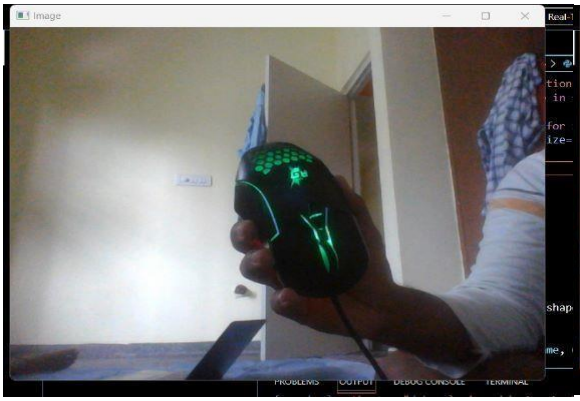

Figure 5.6 Testcase 1 not detecting the object

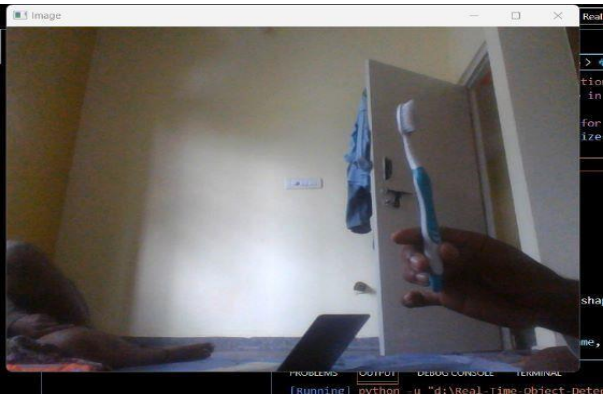

Figure 5.7 Testcase 2 not detecting the object

YOLOv3 is an earlier version of the YOLO (You Only Look Once) family of object detection models. It is known for its abilityto perform real-time object detection at reasonable accuracy. Here are key characteristics and a comparison with YOLOv7:

- YOLOv3 achieves respectable precision in object detection tasks. However, it may struggle with small objects andmay not be as accurate as newer models like YOLOv4 and YOLOv5.
- While YOLOv3 is faster than some earlier models, it may not be suitable for real-time detection on low-end hardwaredue to its computational demands.
- YOLOv3 has a relatively larger model size compared to later YOLO versions, which can impact deployment onresource-constrained devices.
- YOLOv3 is well-documented and has been used extensively in various applications. It can be customized for specifictasks through transfer learning.

In summary, YOLOv7 represents a significant advancement over YOLOv3:

- YOLOv7 offers improved accuracy, especially for small objects.
- YOLOv7 is optimized for speed, making it suitable for real-time applications on various hardware.
- YOLOv7 typically has a smaller model size, which is advantageous for resource-constrained environments.
- YOLOv7 retains the flexibility of previous YOLO versions, making it a popular choice for customization and transferlearning.
- In most cases, YOLOv7 is preferred over YOLOv3 due to its overall better performance and efficiency, especially inreal-time and edge computing scenarios. However, the choice between the two models may also depend on specific use cases and hardware constraints.
- Only 55 out of the 80 things in the coco dataset were discovered by YOLO v3 and 62 out of the 80 objects were found by YOLO v7.YOLO v3 and YOLO v7 are two popular object detection algorithms. YOLO v3 was released in 2018 and YOLO v7 was released in 2023. YOLO v7 is the latest version of YOLO and includes several improvements over YOLO v3, including: Speed:YOLO v7 is significantly faster thanYOLO v3. YOLO v3 can process images at a rate of 155 frames per second, compared to 45 frames per second. Accuracy: YOLO v7 is also more accurate than YOLO v3. We achieve an average accuracy(AP) of 51.4% on the COCO dataset, compared to **45.3%** on YOLO v3. Model size: YOLO v7 is smaller than YOLO v3, so it requiresless memory and storage space. New features: YOLO v7 also includes several new features, including:

Anchor boxes: YOLO v7 uses anchor boxes to detect objects of different shapes and sizes. This allows it to recognize a wider range of objects than YOLO v3. High resolution: YOLO v7 processes images at a resolution of 608 x 608 pixels, which is higher than the 416 x 416 resolution used in YOLO v3. This higher resolution allows YOLO v7 to detect smaller objects and improve overall accuracy. Overall, YOLO v7 is a huge improvement over his YOLO v3. It's faster, more accurate, smaller and has more features analysis.

The main advantage of YOLO v7 over YOLO v3 is its speed. YOLO v7 is significantly faster than

YOLO v3, making it more suitable for real-time applications. For example, YOLO v7 can be used to develop self-driving cars that can detect objects on the road in real time.

Another advantage of YOLO v7 is its accuracy. YOLO v7 is more accurate than YOLO v3. This means it is less likely to miss objects or generate false positives. This is important for applications where accuracy is important, such as medical imaging and security surveillance. YOLO v7 is smaller than YOLO v3, so it requires less memory and storage. This is important for devices with limited resources, such as smartphones and IoT devices.

Finally, YOLO v7 includes several new features such as anchor boxes and higher resolution. These features allow YOLO v7 to detect a wider range of objects and achieve higher overall accuracy. YOLO v7 is a huge improvement over YOLO v3. It's faster, more accurate, smaller and has more features. This makes it a more versatile and powerful object detection algorithm.

7. Conclusion

YOLOv7 is a state-of-the-art object detection model that outperforms YOLOv3 in terms of both speed and accuracy. YOLOv7 achieves this by using a number of innovative techniques, including a new backbone network, a more efficient feature pyramid network, and a new loss function. On the COCO object detection benchmark, YOLOv7 is significantly faster than YOLOv3, while also achieving comparable accuracy. This makes YOLOv7 a superior choice for real-time object detection applications. Overall, YOLOv7 is a better choice than YOLOv3 for real-time object detection applications due to its superior speed and accuracy.

References

[1] .Bharti, C., & Choudhary, R. (2019). Data Preprocessing for Machine Learning A Review.
[2] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once Unified, Real-Time Object Detection.
[3] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD Single Shot MultiBoxDetector.
[4] MobileNetV2: Inverted Residuals and Linear Bottlenecks by Mark Sandler
[5] Tan, M., Pang, R., Le, Q. V., & Vasudevan, V. (2020). EfficientDet Scalable and Efficient Object Detection.
[6] Tong, H., Zhou, K., Fu, H., & Yu, X. (2019). Real-Time Multiple Object Detection on FPGA.
[7] EfficientDet: Scalable and Efficient Object Detection by Mingxing Tan
[8] Real-Time Object Detection with YOLO by Alexey Gruzdev.
[9] Real-Time Object Detection with OpenCV and YOLO by PyImageSearch.
[10] Gupta, A., Vatsa, A. K., & Singh, R. (2020). Static Object Detection in Geo-Spatial Image Using Deep Learning.
[11] Mamatha, N. G., & Giri Prasad, M. N. (2021). Moving Object Detection in Video A Survey.
[12] Dubey, S. R., & Potdar, R. M. (2021). Deep Learning for Visual Object Recognition A Comprehensive Survey.
[13] Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4 Optimal Speed and Accuracy of Object Detection.
[14] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J.,.& Zheng, X. (2016). TensorFlow A system for large- scale machine learning.
[15] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). PyTorch An imperative style, high-performance deep learning library.
[16] Singh, G., & Kaur, S. (2021). A Review of Object Detection Methods and Techniques in the Computer Vision Domain.
[17] Focal Loss for Dense Object Detection by Tsung-Yi Lin
[18] Bredell, G., & van Rooyen, G. J. (2017). Review of Object Detection Performance Metrics and Evaluation Techniques.
[19] Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning.
[20] Zhu, X., Zhang, J., Han, W., & Yu, N. (2021). Deep learning in object detection applications A survey.