**YOLO v4 vs Mask RCNN: A comparative study between two widely used Real-Time object detection techniques.**

**Sameer Ansari1, Tanish Verma2, Shubham Chandrol3**

1 Department of Computer Science, Acropolis Institute of technology & Research,

Indore - 452001, Madhya Pradesh, India.

2 Department of Computer Science, Acropolis Institute of technology & Research,

Indore - 452001, Madhya Pradesh, India.

3 Department of Computer Science, Acropolis Institute of technology & Research,

Indore - 452001, Madhya Pradesh, India.

---------------------------------------------------------------------\*\*\*---------------------------------------------------------------------

**Abstract –** In this work we are going to compare two most widely used real-time object detection algorithms – You Look Only Once (YOLO) and Mask RCNN. YOLO seems to be a

nicer way of doing object detection because it’s fully

end-to-end training but the Mask RCNN too offers end-to-

end training as well, but also along with object detection it also performs semantic segmentation on the image. We will be comparing both of these approaches carefully and in detail.

***Key Words*:** semantic, segmentation, Mask.

**1. INTRODUCTION**

Before starting with comparison between both the techniques, we need to first discuss about object detection. Object detection has been witnessing a huge and rapid change in the field of computer vision.

Its involvement in object classification as well as object detection makes it one of the most popular and challenging topic in the field of computer vision. In other words, the goal of these detection techniques is to determine where objects are located in a given image called as object detection or object localization and which category each object belongs to is called as object classification.

* 1. **You Only Look Once (YOLO)**

YOLO is an algorithm that observes or scans the picture and recognizes various objects in a picture, all this happens in real time. The process of object detection in YOLO is treated as a regression problem and provides the class classification probabilities of the detected images. YOLO uses convolutional neural networks to detect objects in real-time. As the name suggests, the algorithm makes only a single forward propagation through a neural-network to detect objects and the prediction in the entire image is done in a single algorithm run. The convolutional neural network is used to predict various class classification probabilities and bounding boxes simultaneously. The YOLO algorithm consists of various variants. Some of the widely used ones include tiny YOLO v3 and YOLO v4, in this work we will be using YOLO v4.

* 1. **Mask RCNN**

[Mask RCNN](https://openaccess.thecvf.com/content_iccv_2017/html/He_Mask_R-CNN_ICCV_2017_paper.html) is an object detection algorithm based on deep CNN developed by a group of Facebook AI researchers in 2017. The algorithm can return both the bounding box and a mask (color) for each detected object in an image. RCNN in Mask RCNN stands for Region-based Convolutional Neural Network. Mask RCNN extends Faster R-CNN by adding a feature that outputs a binary mask that determines whether or not a given pixel is part of an object. This features is just a Fully Convolutional Network on top of a CNN based feature map.

## Methodology

* 1. **Dataset used – Microsoft COCO**

COCO is a dataset developed by Microsoft and is abbreviated for common objects in context. It is the gold standard benchmark for evaluating the performance of various [computer vision models](https://blog.roboflow.com/yolov5-improvements-and-evaluation/). Despite its amazing features, the [COCO dataset](https://roboflow.com/formats/coco-json) is comparatively less known to general practitioners. Some key facts about COCO dataset –

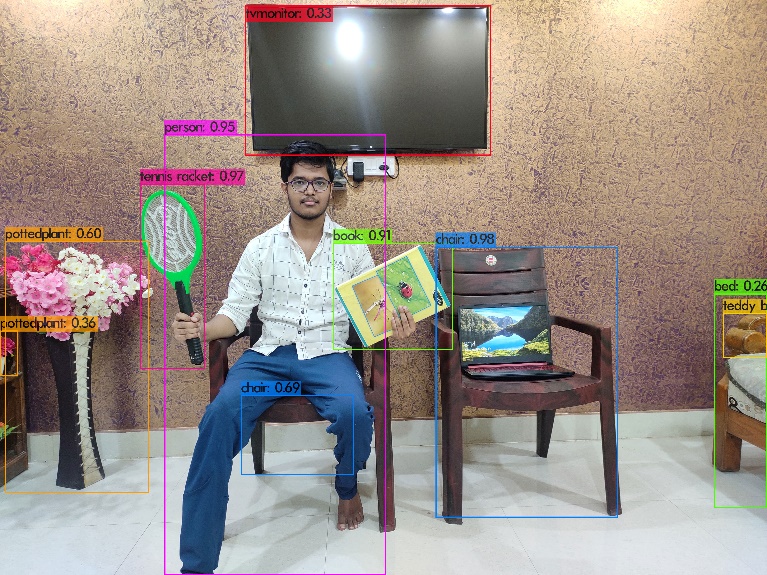
* The COCO Dataset has 121,428 images
* The COCO Dataset has 893,331 object annotations
* The COCO Dataset has 91 classes
* The COCO Dataset image ratio is 640 x 480.
  1. **Study and implementation of YOLO v4**

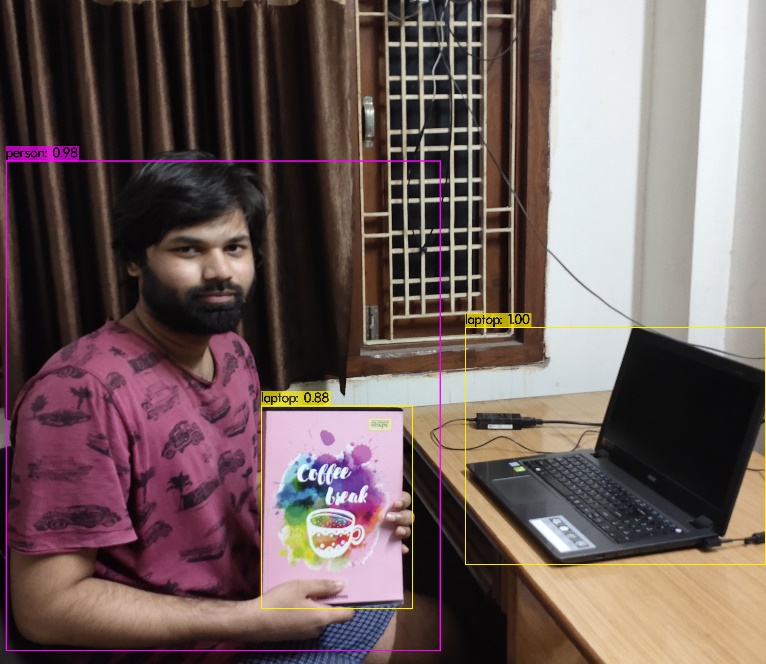
**Backbone**: We have used **CSPDarknet53**as the feature-extractor model, as we used GPU instead of Video Processing Unit.

**Neck**: In Path Aggregation Network, after the size of N4 is reduced to have the same spatial size as P5,this new down-sized N4 added to P5. In YOLO v4 instead of adding 𝑁𝑖 with each 𝑃𝑖+1, we **concatenate**them, this basically performs max-pooling over the feature map with different kernel size*s* and ‘same’ padding. This improves the model accuracy with negligible increase of inference time.

**Head**: The heads applied at different scales of the network, for detecting different-size objects. The number of channels in YOLO v4 is (80 classes + 1 for object + 4 coordinates) \* 3 anchors.

* + 1. **Results from YOLO v4 implementation**





**2.3 Study and implementation of Mask RCNN**

Mask RCNN is an algorithm used for instance

segmentation, it is an extension of Faster RCNN.

Faster RCNN is a region-based convolutional neural

networks, which returns bounding boxes for each

object and also classifies it with a confidence score.

Mask RCNN is basically divided into two parts

Part 1: This part consists of two networks,

backbone and RPN. They run once per

image to give a set of region proposals.

Part 2: In the second part, the network predicts

bounding box and the object classification for each of the proposed regions we got in part 1. Each proposed

region can be of different size whereas fully

connected layers in the networks require

fixed-size vectors to make a prediction. Size of these

proposed regions is fixed by using Region of Interest pool.

**PointRend**

It is a method of instance segmentation proposed by the Facebook AI laboratory. According to them the challenge of image segmentation can also be solved by rendering, and proposed PointRend. This module can be built on various models that currently exist.

In various quantitative evaluations on the COCO dataset, PointRend has achieved significant gains by improving mask AP by 1-2% points for both object detection and semantic segmentation. PointRend also depends on a backbone. The backbone highly influences the segmentation performance.

**Loss function**

Faster RCNN uses cross-entropy for foreground and

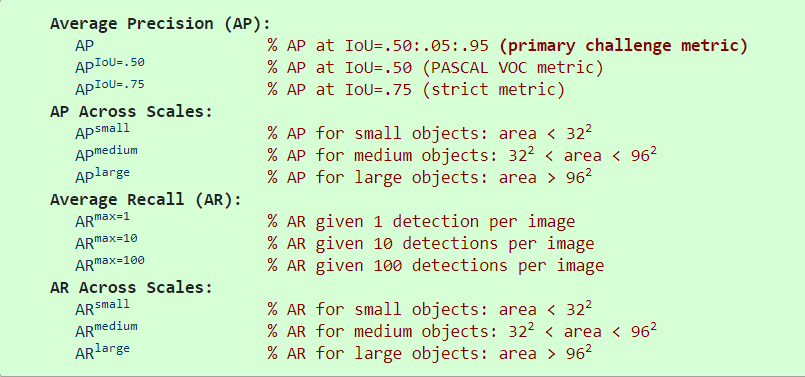
background loss, and l1 regression for coordinates.

**2.3.1 Results from Mask RCNN implementation**

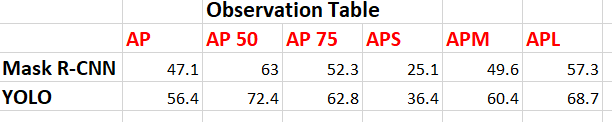


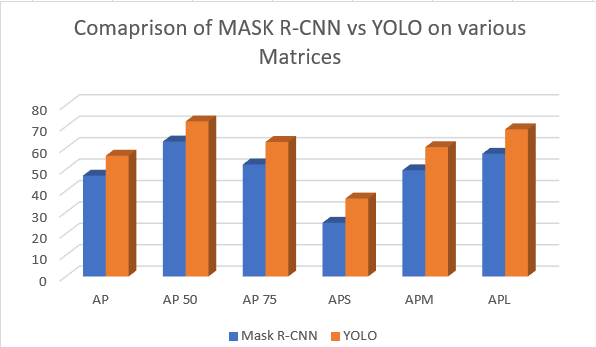


* 1. **Comparison Metrices u**



## Results –





**3. CONCLUSIONS**

In this study we have made the comparison between two of the most widely used Real-time object detection algorithms - Mask RCNN and YOLO using Microsoft COCO dataset as standard benchmark. We can observe from the acquired data or result that under every matrix the YOLO algorithm outperforms Mask RCNN by an average margin of 10.4, which is highly significant with respect to real-time object detection. Thus we can conclude that, on Microsoft COCO dataset, performance of YOLO v4 is superior to that of Mask RCNN.

**REFERENCES**

1. [Kaiming He](https://arxiv.org/search/cs?searchtype=author&query=He%2C+K), [Georgia Gkioxari](https://arxiv.org/search/cs?searchtype=author&query=Gkioxari%2C+G), [Piotr Dollár](https://arxiv.org/search/cs?searchtype=author&query=Doll%C3%A1r%2C+P), [Ross Girshick](https://arxiv.org/search/cs?searchtype=author&query=Girshick%2C+R): Mask RCNN (2017)  [arXiv:1703.06870](https://arxiv.org/abs/1703.06870)

2. [Chien-Yao Wang](https://arxiv.org/search/cs?searchtype=author&query=Wang%2C+C), [Alexey Bochkovskiy](https://arxiv.org/search/cs?searchtype=author&query=Bochkovskiy%2C+A), [Hong-YuanMark Liao](https://arxiv.org/search/cs?searchtype=author&query=Liao%2C+H+M): Scaled-YOLOv4: Scaling Cross Stage PartialNetwork (2020) [arXiv:2011.08036](https://arxiv.org/abs/2011.08036).

3. [Zhong-Qiu Zhao](https://arxiv.org/search/cs?searchtype=author&query=Zhao%2C+Z), [Peng Zheng](https://arxiv.org/search/cs?searchtype=author&query=Zheng%2C+P), [Shou-tao Xu](https://arxiv.org/search/cs?searchtype=author&query=Xu%2C+S), [Xindong Wu](https://arxiv.org/search/cs?searchtype=author&query=Wu%2C+X): Object Detection with Deep Learning: A Review (2019),  [arXiv:1807.05511](https://arxiv.org/abs/1807.05511).