UNIVERSITY OF TEHRAN

COLLEGE OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NEURAL NETWORK & DEEP LEARNING

Object Detection Using YOLOv5

SIAVASH SHAMS

MOHAMMAD HEYDARI

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

UNIVERSITY OF TEHRAN

Apr. 2022

1 CONTENTS

Object Detection **3**
YOLOv2 Vs YOLOv1 3
YOLOv5 and YOLOv4 Advances 4
One stage Vs Two Stage Object Detection 5
YOLOv5 for Bocce Balls 6

OBJECT DETECTION

YOLOv2 VS YOLOv1

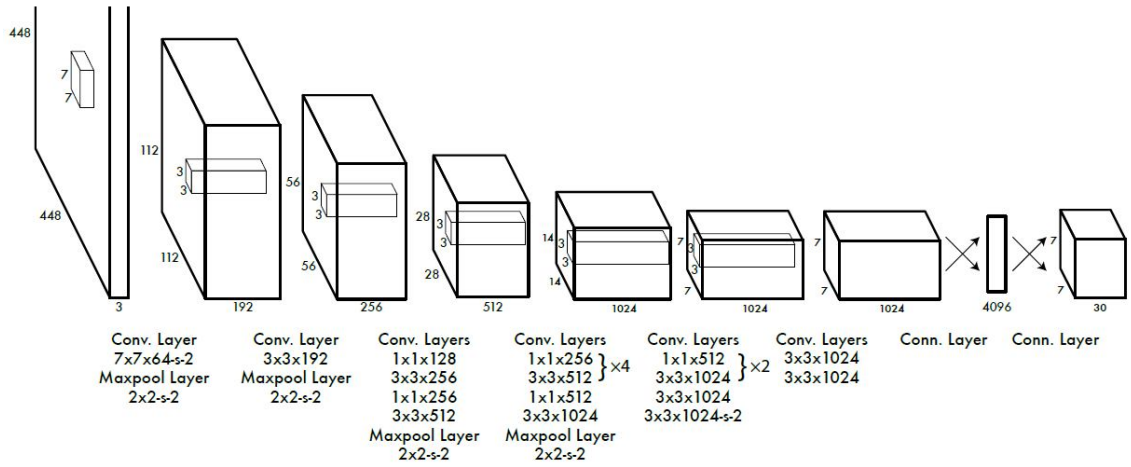

Figure31. Yolov1

The changes from YOLOv2 to YOLO v1:

- **Batch Normalization:** it normalizes the input layer by altering slightly and scaling the activations. Batch normalization decreases the shift in unit value in the hidden layer and by doing so it improves the stability of the neural network.
- **Higher Resolution Classifier**: the input size in YOLO v2 has been increased from 224*224 to 448*448.
- **Anchor Boxes**: one of the most notable changes which can visible in YOLO v2 is the introduction the anchor boxes. YOLO v2 does classification and prediction in a single framework. These anchor boxes are responsible for predicting bounding box and this anchor boxes are designed for a given dataset by using clustering (k-means clustering)
- **Fine-Grained Features**: one of the main issued that has to be addressed in the YOLO v1 is that detection of smaller objects on the image. This has been resolved in the YOLO v2 divides the image into 13*13 grid cells which is smaller when compared to its previous version. This enables the yolo v2 to identify or localize the smaller objects in the image and also effective with the larger objects.
- **Multi-Scale Training**: on YOLO v1 has a weakness detecting objects with different input sizes which says that if YOLO is trained with small images of a particular object. it has issues detecting the same object on image of bigger size.
- **Darknet 19**: YOLO v2 uses Darknet 19 architecture with 19 convolutional layers and 5 max pooling layers and a softmax layer for classification objects. The architecture of the Darknet 19 has been shown below. Darknet is a neural network framework written in Clanguage and CUDA. It's really fast in object detection which is very important for predicting in real-time.

Table1. Darknet 19 architecture

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

YOLOv2 has seen a great improvement in detecting smaller objects with much more accuracy which it lacked in its predecessor version.

YOLOv5 AND YOLOv4 ADVANCES

YOLO4. uses Cross Stage Partial Network (CSPNet) in Darknet, creating a new feature extractor backbone called CSPDarknet53. The convolution architecture is based on modified DenseNet. It transfers a copy of feature map from the base layer to the next layer through dense block. The advantages of using DenseNet include the diminishing gradient vanishing problems, boosting backpropagation, removal of the computational bottleneck, and improved learning.

However, YOLOv5 is different from the previous releases. It utilizes PyTorch instead of Darknet. It utilizes CSPDarknet53 as backbone. This backbone solves the repetitive gradient information in large backbones and integrates gradient change into feature map that reduces the inference speed, increases accuracy, and reduces the model size by decreasing the parameters.

Table2.

	YOLOv3	YOLOv4	YOLOv5
Neural Network Type	Fully convolution	Fully convolution	Fully convolution
Backbone Feature Extractor	Darknet-53	CSPDarknet53	CSPDarknet53
Loss Function	Binary cross entropy	Binary cross entropy	Binary cross entropy and Logits loss function
Neck	FPN	SSP and PANet	PANet
Head	YOLO layer	YOLO layer	YOLO layer

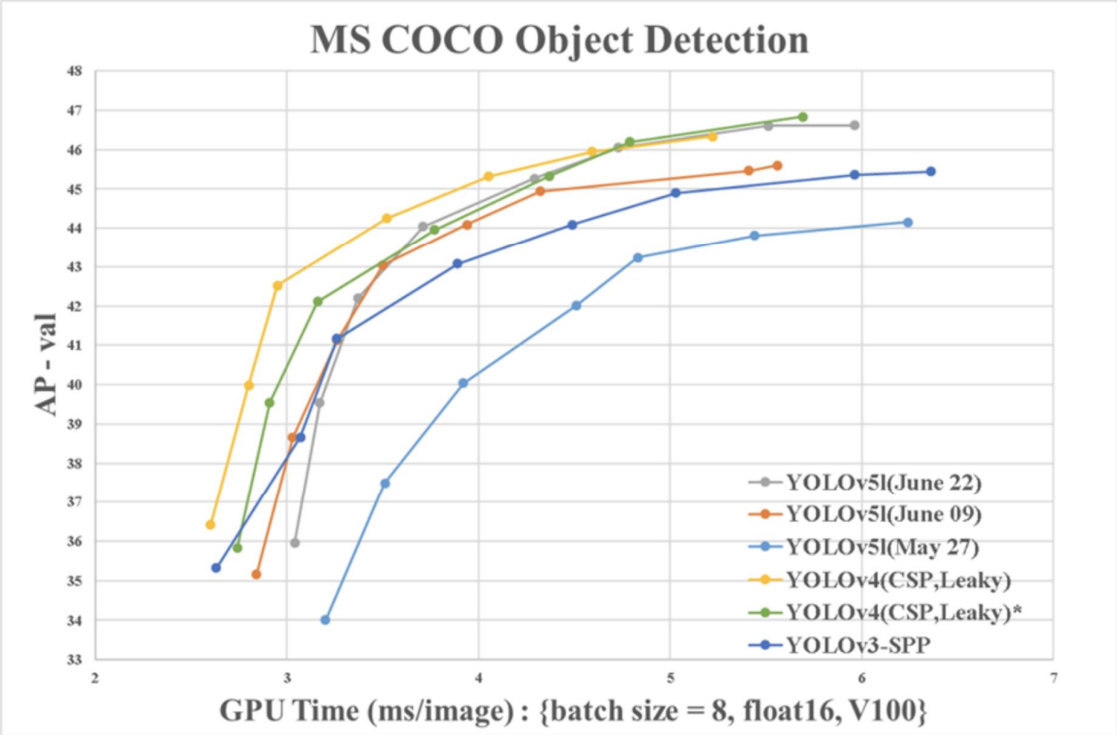

Figure32. Average precision against GPU Time for different versions of YOLO

ONE STAGE VS TWO STAGE OBJECT DETECTION

One-stage detectors have high inference speeds and two-stage detectors have high localization and recognition accuracy. The two stages of a two-stage detector can be divided by a RoI (Region of Interest) Pooling layer. One of the prominent two-stage object detectors is Faster R-CNN. It has the first stage called RPN, a Region Proposal Network to predict candidate bounding boxes. In

the second stage, features are by RoI pooling operation from each candidate box for the following classification and bounding-box regression tasks. In contrast, a one-stage detector predicts bounding boxes in a single-step without using region proposals. It leverages the help of a grid box and anchors to localize the region of detection in the image and constraint the shape of the object.

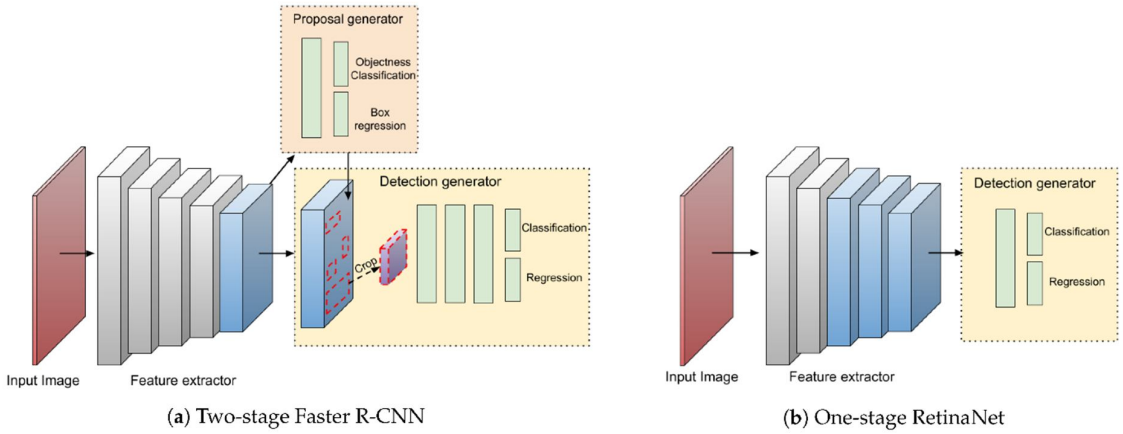

Figure33. Average precision against GPU Time for different versions of YOLO

Some of one-stage object detection methods are: YOLO, EfficientDet, RefineDet

And for two-stage object detection methods we can name: Fast R-CNN, PANet

YOLOv5 FOR BOCCE BALLS

We trained our YOLOv5 model for the Booceball dataset which has 6 objects to detect; Balls with 5 different colors and vlines.

Training time: 1hour 28minutes on Tesla K80 GPU and 12GB of RAM, 100 epochs

First, we upload the dataset with a .yaml file on the RoboFlow website and then use the roboflow library in python with the API obtained from the website to load the data in the Colab environment. Then we git clone the YOLOv5 repository in Github and build our model (anchors, backbone, head). Then we train our model on the dataset for 100 epochs using 16 batch sizes. After training, we will show precision and accuracy graphs for validation and training data. Then we will show an augmented (the augmentation procedure is explained in the "README.roboflow.txt") image from the training dataset and a ground truth image with labels from the validation dataset along with the image with the predicted labels by the model. Then we will run the "detect.py" from the YOLOv5 repository to perform object detection on the test dataset and show the results.

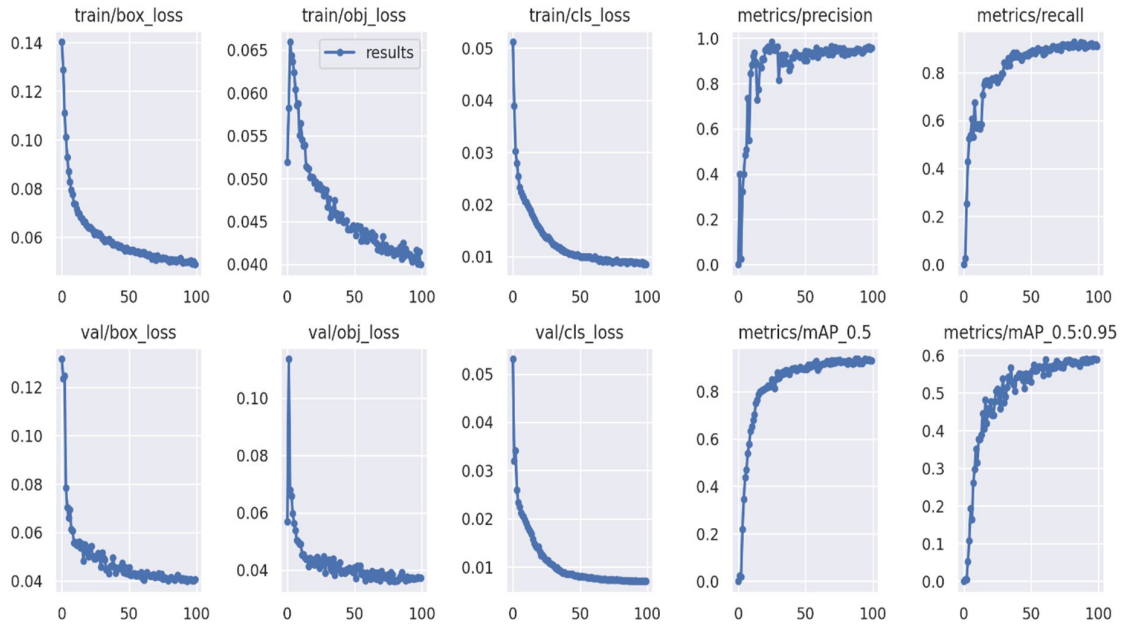

Figure34. Average precision against GPU Time for different versions of YOLO

Figure35. Average precision against GPU Time for different versions of YOLO

Figure36. Average precision against GPU Time for different versions of YOLO

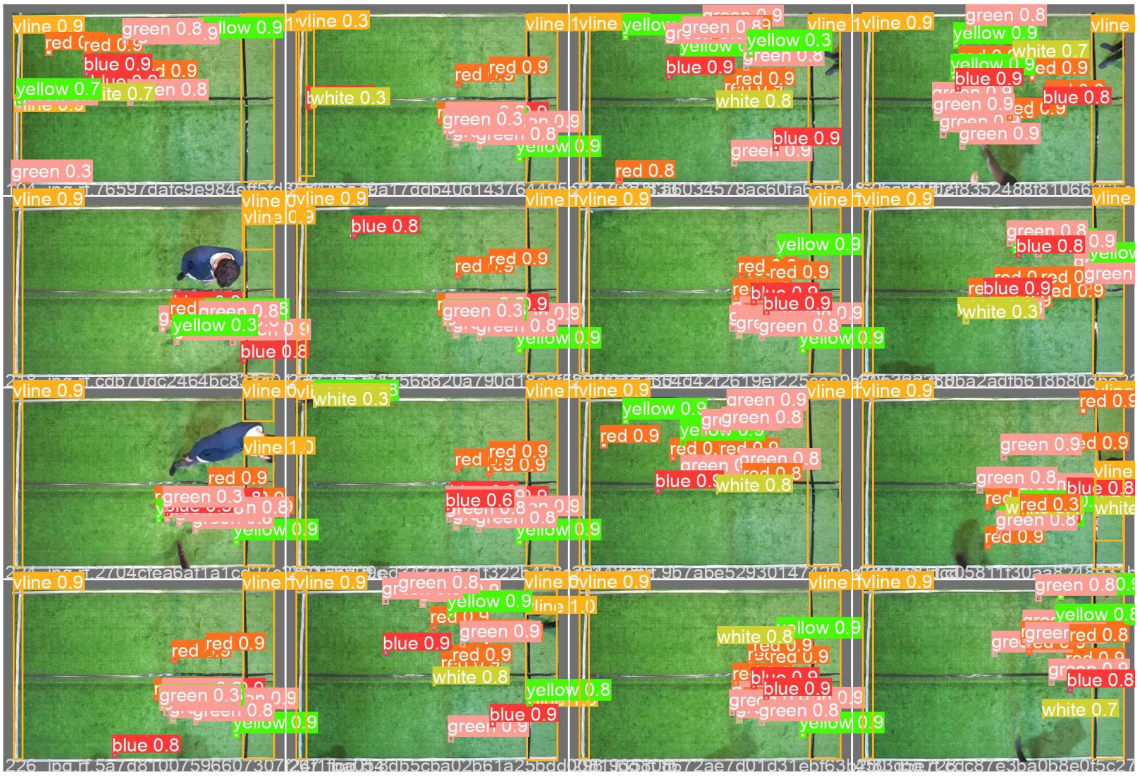

Figure37. Average precision against GPU Time for different versions of YOLO

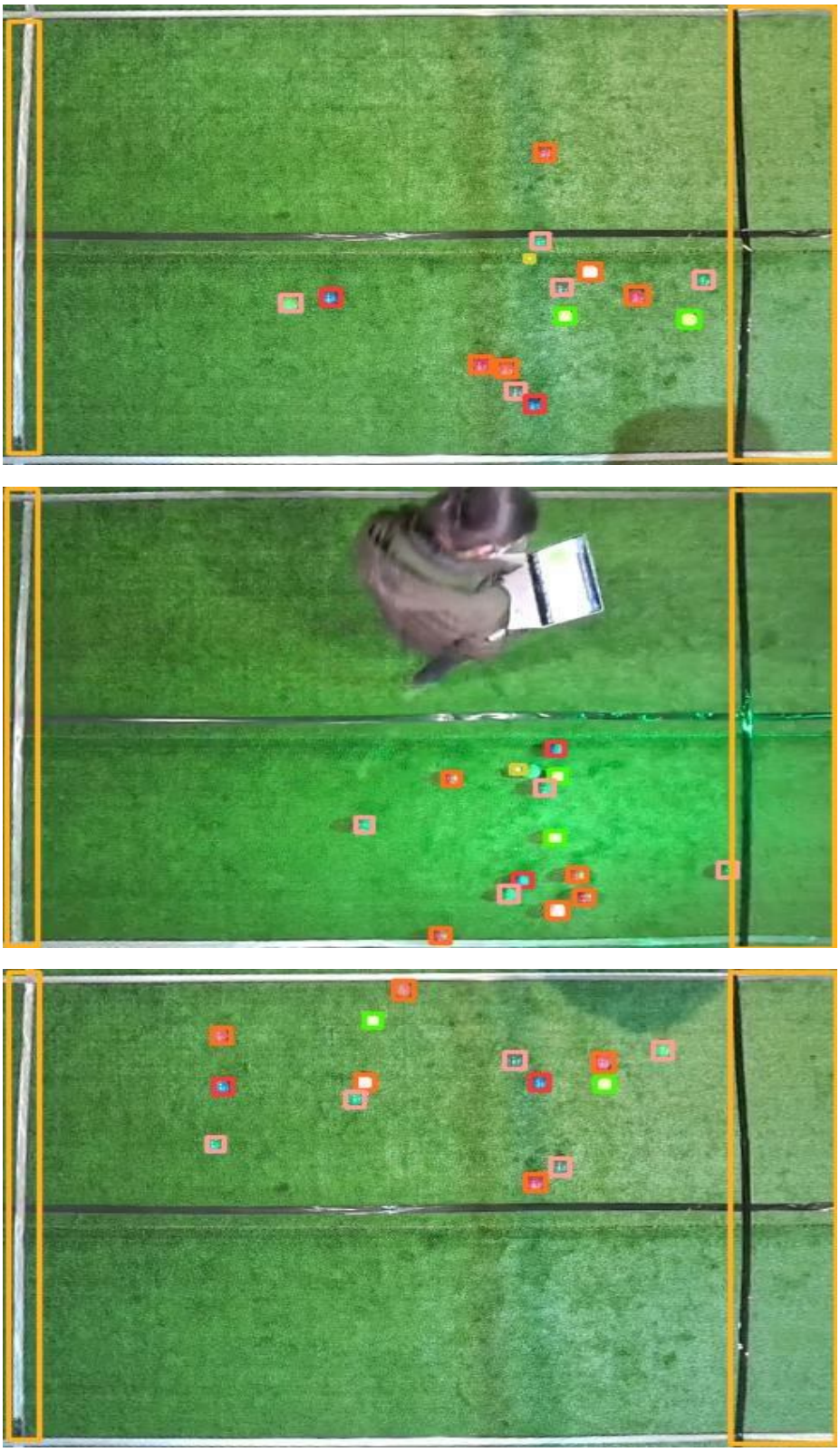

Figure38. Average precision against GPU Time for different versions of YOLOv5