YOLOv7

Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors

PRESENTATION - 2

Presented by :

RAKSHITH RAM C.A.

SM22MTECH12003

Faculty Incharge :

Prof. C. KRISHNA MOHAN

Teaching Assistant :

AVEEN DAYAL

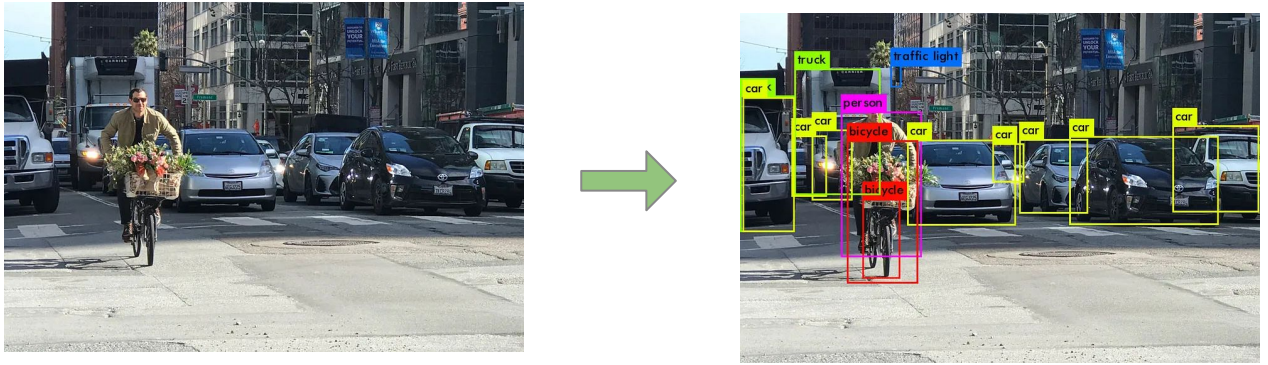
IIT Hyderabad

IIT Hyderabad

“ **YOLOv7** : Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. ”

ArXiv (open access archive), July 2022.

Authors : Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao



- Problem statement
- Applications
- Summary of Presentation - 1
  - Object detection procedure in YOLO
  - Architecture
  - Various concepts used in YOLO
  - Evaluation metrics
  - Experiments and results
- Code Implementation
- Training, Testing and Inference on custom dataset
- Novel idea for future work
- References

Problem statement

1. Achieve state-of-the-art performance → High accuracy and low false positive rates.

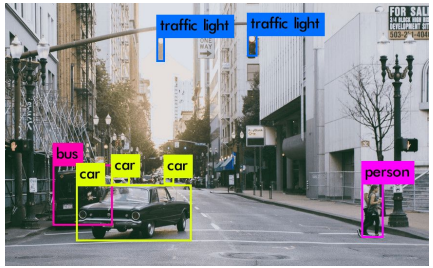
1. Process images in real-time → Suitable for applications such as autonomous driving, surveillance, etc.

1. Be efficient in terms of computational resources → Memory and processing power.

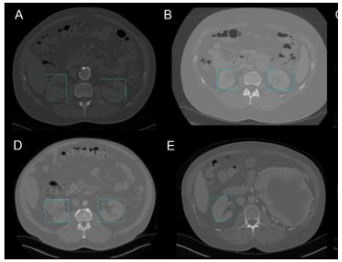
Applications of object detection and YOLO

IIT Hyderabad

1. Self-driving cars
2. Traffic monitoring
3. Parking occupancy
4. Vehicle tracking
5. Healthcare
6. Agriculture
7. Animal detection in agriculture
8. PPE detection
9. Inventory management
10. Contactless checkout
11. Defect and anomaly detection
12. Video analytics
13. Security surveillance
14. Face detection

a. 

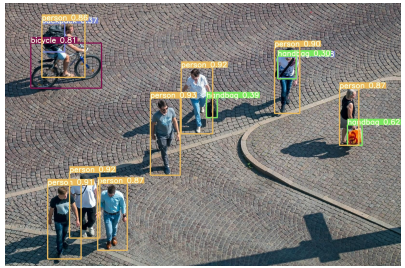
a) Self-driving cars

b. 

b) MRI scans

c. 

c) Agriculture

d. 

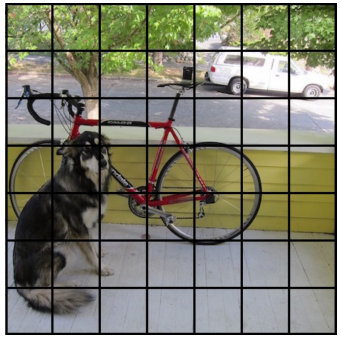
d) Security surveillance

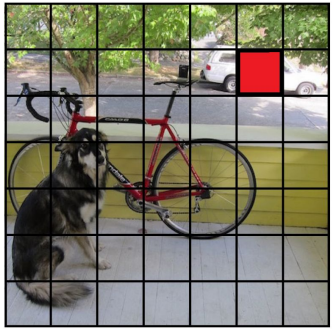
Object detection procedure in YOLO

Input image

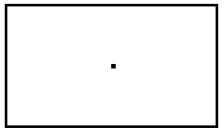
The image is split into **S * S** grid

here , **S** = 7





Each cell of the grid predicts **B** boxes **(x,y,w,h)** and the confidences of each of the box **P (object)**

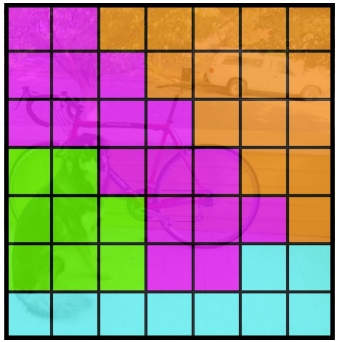


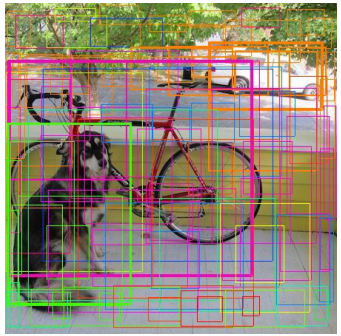
here, **B** = 2

All of the cells predict boxes and confidences.

Each of the cell also predicts a class probability.

- Car
- Bicycle
- Dog

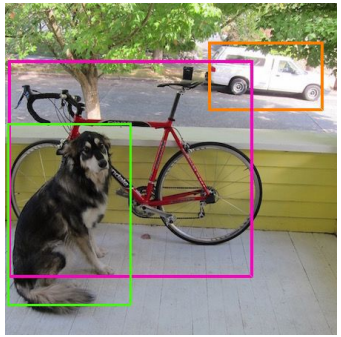




The output bounding boxes and the class predictions are combined.

$$P(\text{class} \mid \text{object}) \times P(\text{object}) = P(\text{class})$$

Finally, **NMS** is performed to obtain the best bounding boxes.



Each cell predicts :

For each bounding box :

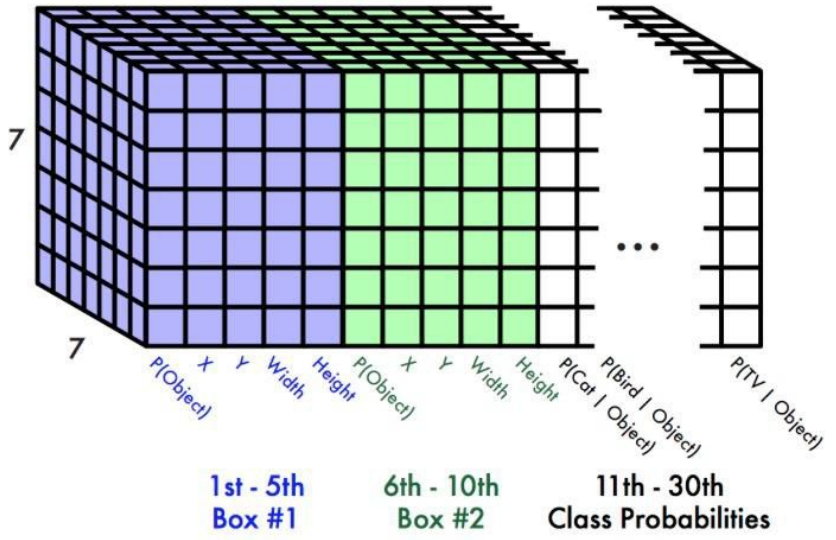
- 4 co-ordinates (x, y, w, h)
- 1 confidence value
- Some number of class probabilities

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Example :

- 7 * 7 grid
- 2 bounding boxes
- 20 classes

7 * 7 * (2 * 5 +20) = 1470 outputs

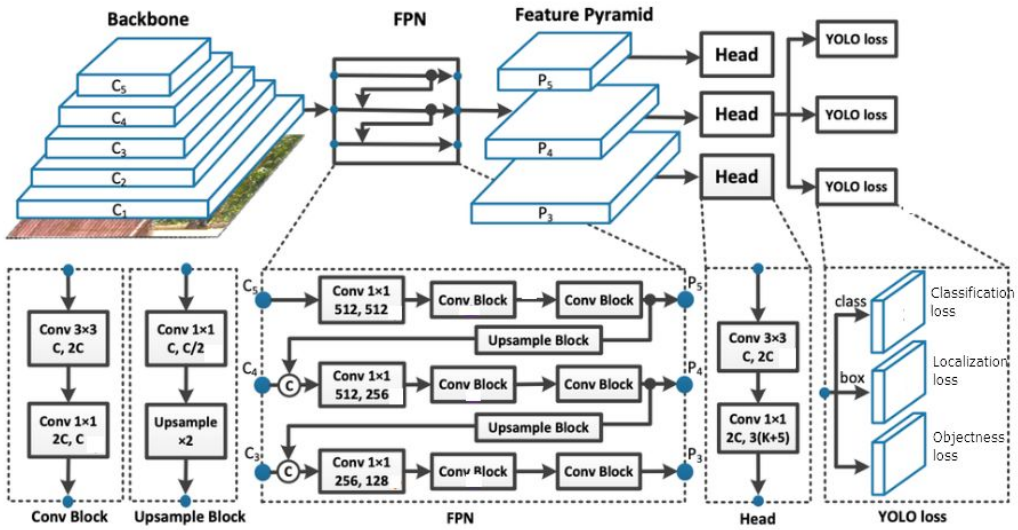


YOLOv7 - Architecture

IIT Hyderabad

The YOLO framework has three main components :

- ➔ The **Backbone** mainly extracts essential features of an image and feeds them to the Neck.
- ➔ The **Neck** collects feature maps extracted by the Backbone and creates feature pyramids.
- ➔ Finally, the **Head** consists of output layers that have final detections.



Loss functions in YOLOv7

IIT Hyderabad

The YOLOv7 model uses three main loss functions to optimize the model for object detection:

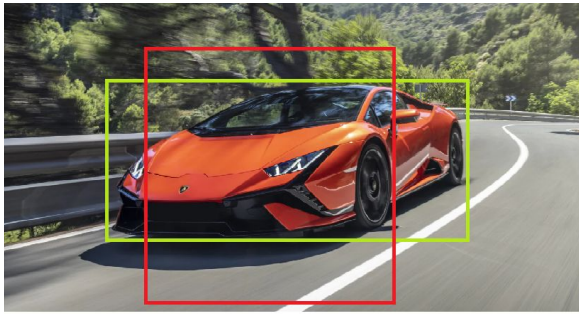
1. **Objectness loss:** This loss function is used to train the model to predict whether an object is present in a given bounding box. The objectness loss is computed using a binary cross-entropy loss between the predicted objectness score and the ground truth objectness score.

1. **Classification loss:** The classification loss is computed using a multi-class cross-entropy loss between the predicted class probabilities and the ground truth class labels.

1. **Localization loss:** The localization loss is computed using the mean squared error (MSE) between the predicted bounding box coordinates and the ground truth bounding box coordinates.

Intersection over Union (IoU)

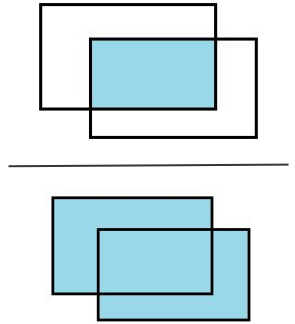
Intersection over Union is a popular metric to measure localization accuracy and calculate localization errors in object detection models.



Predicted bounding box

Ground truth

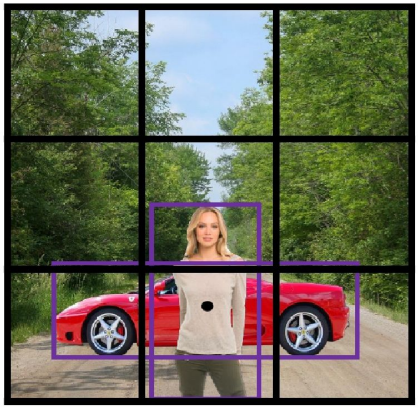
$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}} =$$



Higher the IoU value, closer the bounding box is to the ground truth.

Generally a threshold (example : IoU > 0.5) is set.

Anchor boxes



Anchor box 1

Anchor box 2

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

- The concept of the Anchor box is to pre-define multiple Anchor boxes of different shapes, and then combine the prediction results with multiple Anchor boxes.
- For several Anchor boxes, the vector of the output y will be stretched several times for correlation.

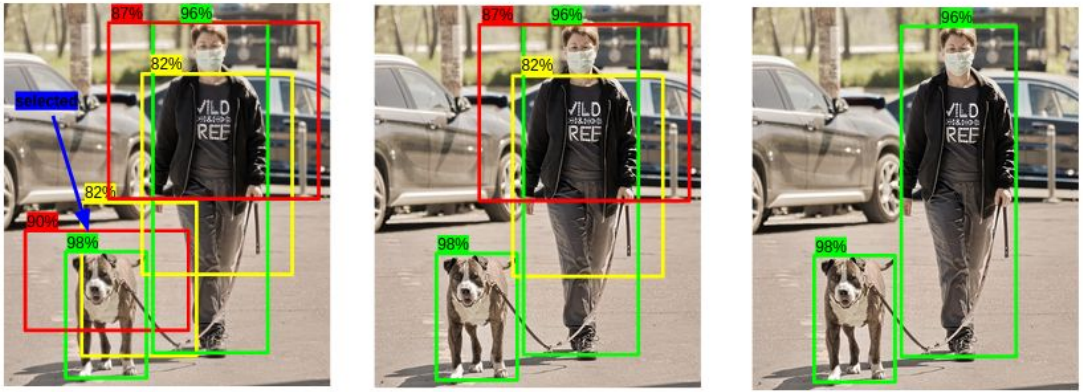
➔ YOLOv5 and YOLOv7 use **9** anchor boxes by default.

Non-maximum suppression (NMS)

The purpose of non-max suppression is to select the best bounding box for an object and reject or “suppress” all other bounding boxes.

The NMS takes two things into account :

1. The objectiveness score is given by the model.
2. The overlap or IOU of the bounding boxes.



Evaluation metrics

AP = Area under the Precision-Recall curve

$$\mathrm{AP} = \int_0^1 p(r)\,\mathrm{d}r$$

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

AP_k = the average precision of class k
n = number of classes

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1 score} = \frac{2}{1/\text{Precision} + 1/\text{Recall}}$$

Comparison of baseline object detectors.

Model	#Param.	FLOPs	Size	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
YOLOv4 [3]	64.4M	142.8G	640	49.7%	68.2%	54.3%	32.9%	54.8%	63.7%
YOLOR-u5 (r6.1) [81]	46.5M	109.1G	640	50.2%	68.7%	54.6%	33.2%	55.5%	63.7%
YOLOv4-CSP [79]	52.9M	120.4G	640	50.3%	68.6%	54.9%	34.2%	55.6%	65.1%
YOLOR-CSP [81]	52.9M	120.4G	640	50.8%	69.5%	55.3%	33.7%	56.0%	65.4%
YOLOv7	36.9M	104.7G	640	**51.2%**	**69.7%**	**55.5%**	**35.2%**	**56.0%**	**66.7%**
improvement	-43%	-15%	-	+0.4	+0.2	+0.2	+1.5	=	+1.3
YOLOR-CSP-X [81]	96.9M	226.8G	640	52.7%	**71.3%**	57.4%	36.3%	57.5%	68.3%
YOLOv7-X	71.3M	189.9G	640	**52.9%**	71.1%	**57.5%**	**36.9%**	**57.7%**	**68.6%**
improvement	-36%	-19%	-	+0.2	-0.2	+0.1	+0.6	+0.2	+0.3
YOLOv4-tiny [79]	6.1	6.9	416	24.9%	42.1%	25.7%	8.7%	28.4%	39.2%
YOLOv7-tiny	6.2	5.8	416	**35.2%**	**52.8%**	**37.3%**	**15.7%**	**38.0%**	**53.4%**
improvement	+2%	-19%	-	+10.3	+10.7	+11.6	+7.0	+9.6	+14.2
YOLOv4-tiny-3l [79]	8.7	5.2	320	30.8%	47.3%	32.2%	**10.9%**	31.9%	51.5%
YOLOv7-tiny	6.2	3.5	320	**30.8%**	**47.3%**	**32.2%**	10.0%	**31.9%**	**52.2%**
improvement	-39%	-49%	-	=	=	=	-0.9	=	+0.7
YOLOR-E6 [81]	115.8M	683.2G	1280	55.7%	73.2%	60.7%	40.1%	**60.4%**	69.2%
YOLOv7-E6	97.2M	515.2G	1280	**55.9%**	**73.5%**	**61.1%**	**40.6%**	60.3%	**70.0%**
improvement	-19%	-33%	-	+0.2	+0.3	+0.4	+0.5	-0.1	+0.8
YOLOR-D6 [81]	151.7M	935.6G	1280	56.1%	73.9%	61.2%	**42.4%**	60.5%	69.9%
YOLOv7-D6	154.7M	806.8G	1280	56.3%	73.8%	61.4%	41.3%	60.6%	70.1%
YOLOv7-E6E	151.7M	843.2G	1280	**56.8%**	**74.4%**	**62.1%**	40.8%	**62.1%**	**70.6%**
improvement	=	-11%	-	+0.7	+0.5	+0.9	-1.6	+1.6	+0.7

Code Implementation

IIT Hyderabad

Downloading all the required codes :-

```
git clone https://github.com/WongKinYiu/yolov7.git
%cd yolov7
```

Installing all the requirements :-

```
pip install -r requirements.txt
```

Downloading the required weights :-

```
wget https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7.pt
```

`yolov7.pt` `yolov7x.pt` `yolov7-w6.pt` `yolov7-e6.pt` `yolov7-d6.pt` `yolov7-e6e.pt`

IIT Hyderabad

Inference on an image :-

```
python detect.py --weights yolov7.pt --conf 0.25 --img-size 640 --source inference/images/picture.jpg
```

Inference on an video :-

```
python detect.py --weights yolov7.pt --conf 0.25 --img-size 640 --source inference/images/video.mp4
```

Inference on live webcam :-

```
python detect.py --weights yolov7.pt --conf 0.25 --img-size 640 --source 0
```

Code Implementation

IIT Hyderabad

Downloading all the required codes :-

```
git clone https://github.com/WongKinYiu/yolov7.git
%cd yolov7
```

Installing all the requirements :-

```
pip install -r requirements.txt
```

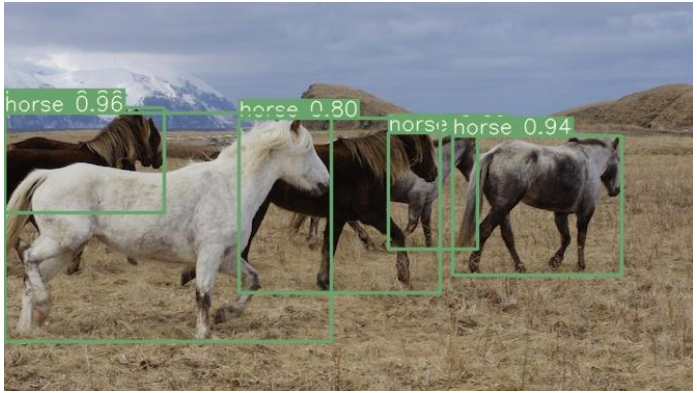
Downloading the required weights :-

```
wget https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7.pt
```

`yolov7.pt` `yolov7x.pt` `yolov7-w6.pt` `yolov7-e6.pt` `yolov7-d6.pt` `yolov7-e6e.pt`


bus 0.96
person 0.91
cero emisiones


tv 0.80
person 0.95
person 0.66
cell phone 0.75
laptop 0.94
cup 0.80

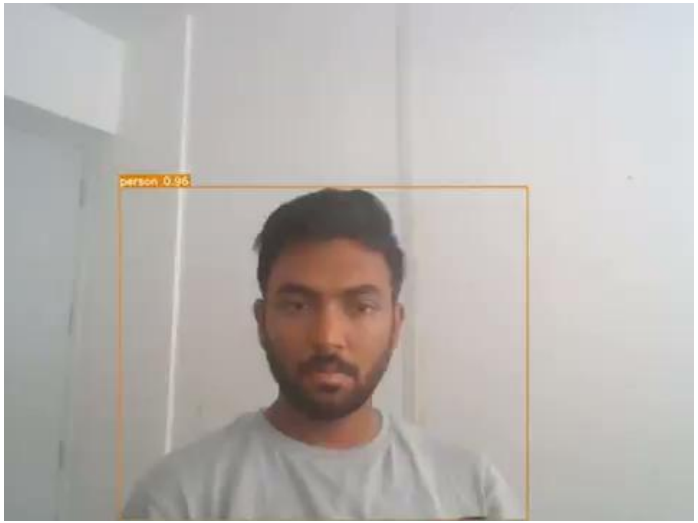

horse 0.96
horse 0.80
horse 0.94


person 0.97
person 0.95
sports ball 0.96
Acronis

Inference on an video :-

IIT Hyderabad

Inference on live webcam :-



IIT Hyderabad

Confidence = 0.6

Confidence = 0.4

IIT Hyderabad

Comment Share R

RAM Disk

detect.py

```python
import argparse
import time
from pathlib import Path

import cv2
import torch
import torch.backends.cudnn as cudnn
from numpy import random

from models.experimental import attempt_load
from utils.datasets import LoadStreams, LoadImages
from utils.general import check_img_size, check_requirements, check_imshow, non_max_suppressior
    scale_coords, xyxy2xywh, strip_optimizer, set_logging, increment_path
from utils.plots import plot_one_box
from utils.torch_utils import select_device, load_classifier, time_synchronized, TracedModel

random.seed(10)  # added by Rakshith

def detect(save_img=False):
    source, weights, view_img, save_txt, imgsz, trace = opt.source, opt.weights, opt.view_img,
    save_img = not opt.nosave and not source.endswith('.txt')  # save inference images
    webcam = source.isnumeric() or source.endswith('.txt') or source.lower().startswith(
        ('rtsp://', 'rtmp://', 'http://', 'https://'))

    # Directories
    save_dir = Path(increment_path(Path(opt.project) / opt.name, exist_ok=opt.exist_ok))  # inc
    (save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True)  # make

    # Initialize
```

RAM Disk

detect.py

```python
                if save_txt:  # Write to file
                    xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist()  # normal
                    line = (cls, *xywh, conf) if opt.save_conf else (cls, *xywh)  # label format
                    with open(txt_path + '.txt', 'a') as f:
                        f.write(('%g ' * len(line)).rstrip() % line + '\n')

                if save_img or view_img:  # Add bbox to image
                    label = f'{names[int(cls)]} {conf:.2f}'
                    plot_one_box(xyxy, im0, label=label, color=colors[int(cls)], line_thickness=2)

        # Print time (inference + NMS)
        print(f'{s}Done. ({(1E3 * (t2 - t1)):.1f}ms) Inference, ({(1E3 * (t3 - t2)):.1f}ms) NMS')

        # Stream results
        if dataset.mode != "image":          #1
          CurrentTime = time.time()          #2

          fps = 1/(CurrentTime - StartTime)  #3
          StartTime = CurrentTime            #4

          cv2.putText(im0, "FPS : " + str(int(fps)), (20, 70), cv2.FONT_HERSHEY_PLAIN, 2, (0,225,0), 2)

        if view_img:
            cv2.imshow(str(p), im0)
            cv2.waitKey(1)  # 1 millisecond

        # Save results (image with detections)
        if save_img:
            if dataset.mode == 'image':
                cv2.imwrite(save_path, im0)
                #print(f" The image with the result is saved in: {save_path}")
            else:  # 'video' or 'stream'
```

Training on Pascal VOC dataset.

IIT Hyderabad

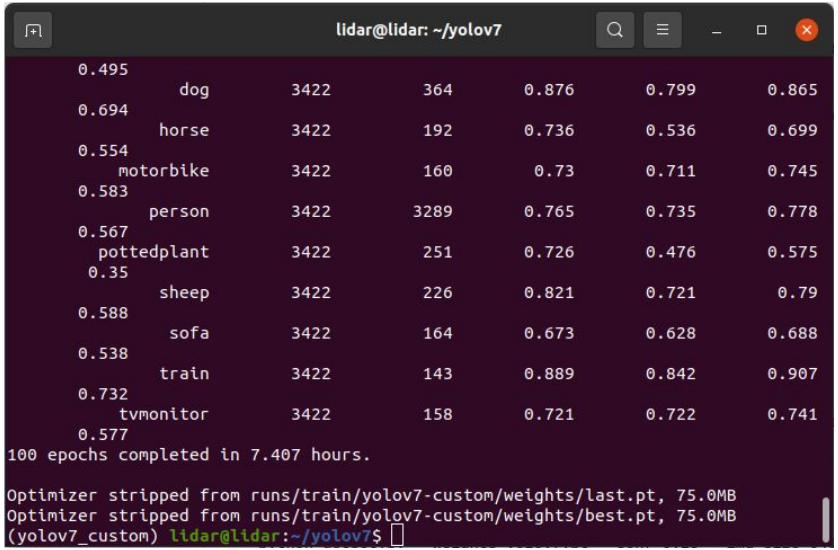
- Around 18,000 images for training, validation and testing.

- *Person:* person

  Animal: bird, cat, cow, dog, horse, sheep

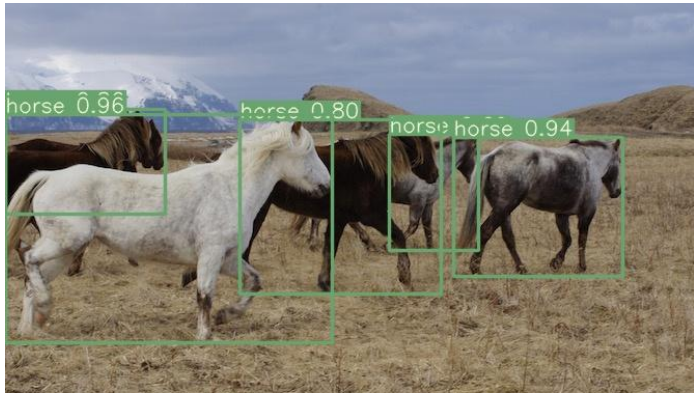
  Vehicle: aeroplane, bicycle, boat, bus, car, motorbike, train

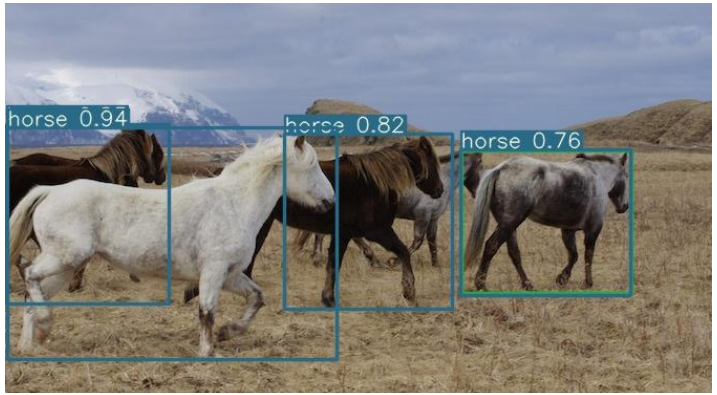
  Indoor: bottle, chair, dining table, potted plant, sofa, tv/monitor

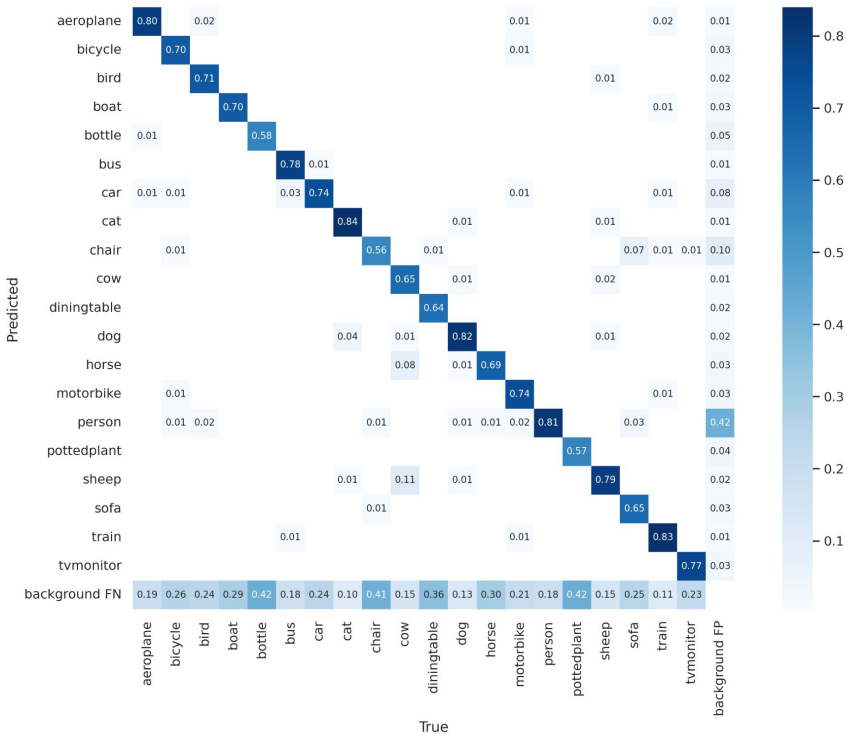



person 0.97
person 0.95
sports ball 0.96

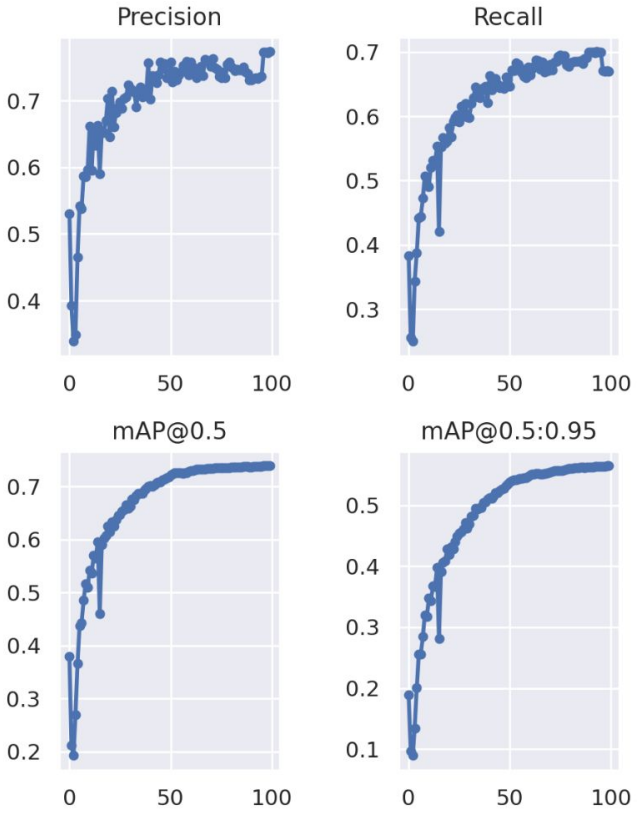

person 0.38
person 0.45

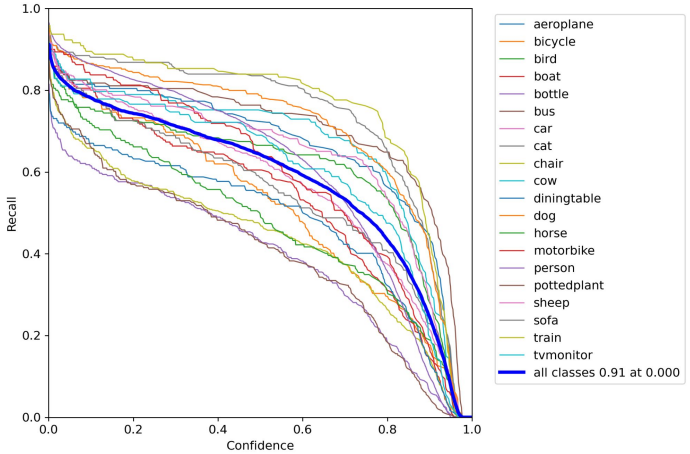

horse 0.96
horse 0.80
horse 0.94

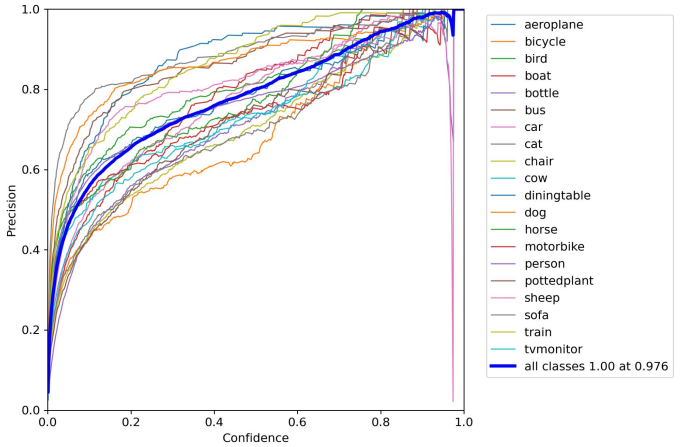

horse 0.94
horse 0.82
horse 0.76

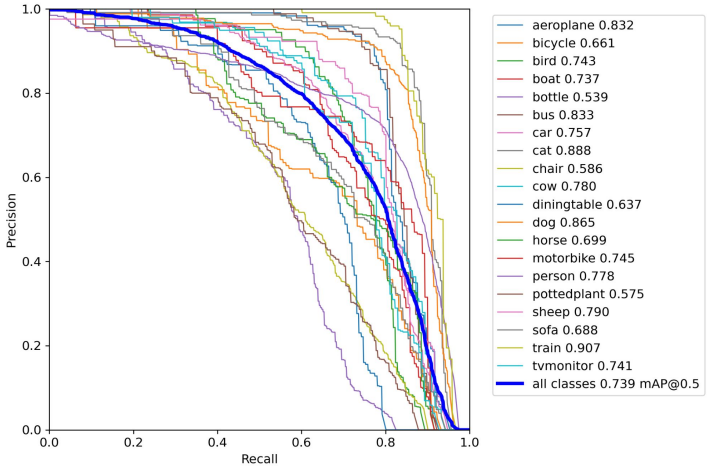


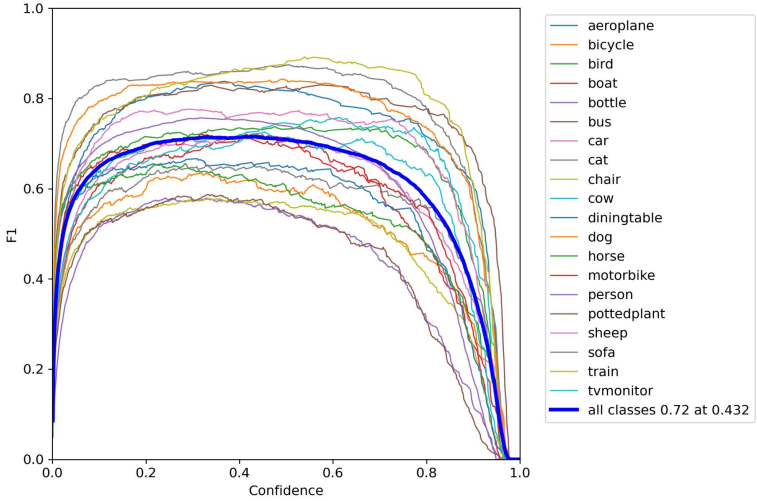
Confusion matrix




Recall
Confidence
aeroplane
bicycle
bird
boat
bottle
bus
car
cat
chair
cow
diningtable
dog
horse
motorbike
person
pottedplant
sheep
sofa
train
tvmonitor
all classes 0.91 at 0.000


Precision
Confidence
aeroplane
bicycle
bird
boat
bottle
bus
car
cat
chair
cow
diningtable
dog
horse
motorbike
person
pottedplant
sheep
sofa
train
tvmonitor
all classes 1.00 at 0.976


Precision
Recall
aeroplane 0.832
bicycle 0.661
bird 0.743
boat 0.737
bottle 0.539
bus 0.833
car 0.757
cat 0.888
chair 0.586
cow 0.780
diningtable 0.637
dog 0.865
horse 0.699
motorbike 0.745
person 0.778
pottedplant 0.575
sheep 0.790
sofa 0.688
train 0.907
tvmonitor 0.741
all classes 0.739 mAP@0.5

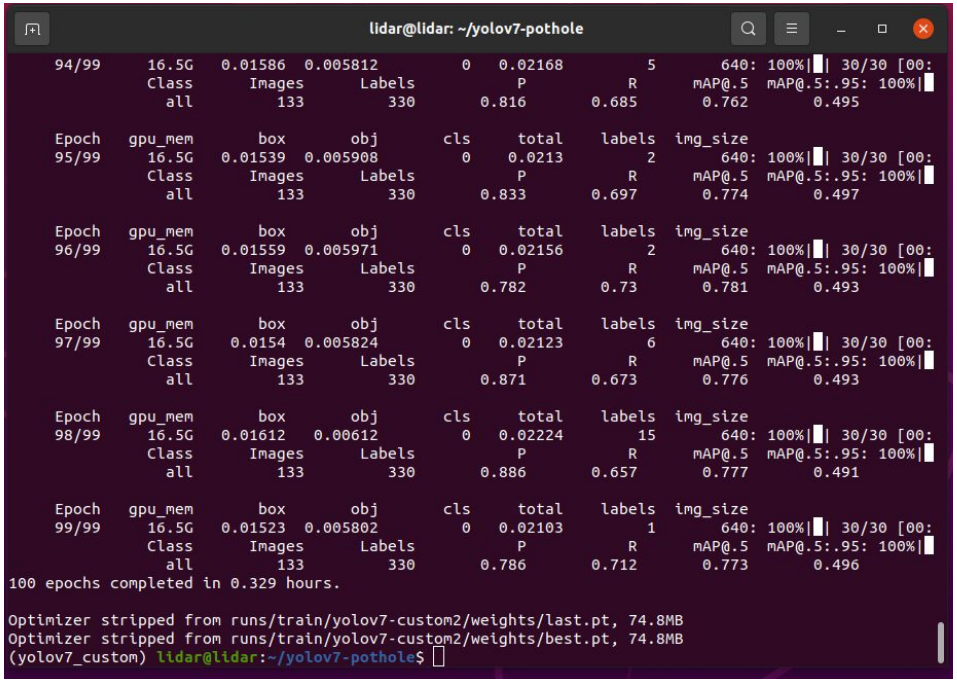

F1
Confidence
aeroplane
bicycle
bird
boat
bottle
bus
car
cat
chair
cow
diningtable
dog
horse
motorbike
person
pottedplant
sheep
sofa
train
tvmonitor
all classes 0.72 at 0.432

Training and Inference on Pothole dataset.

IIT Hyderabad

- Training images : 465
- Validation images : 133
- Test images : 67

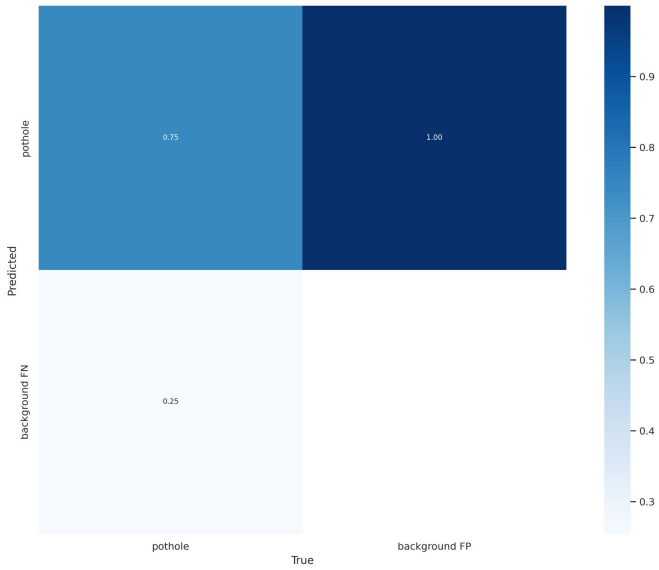
* public dataset downloaded from Roboflow platform



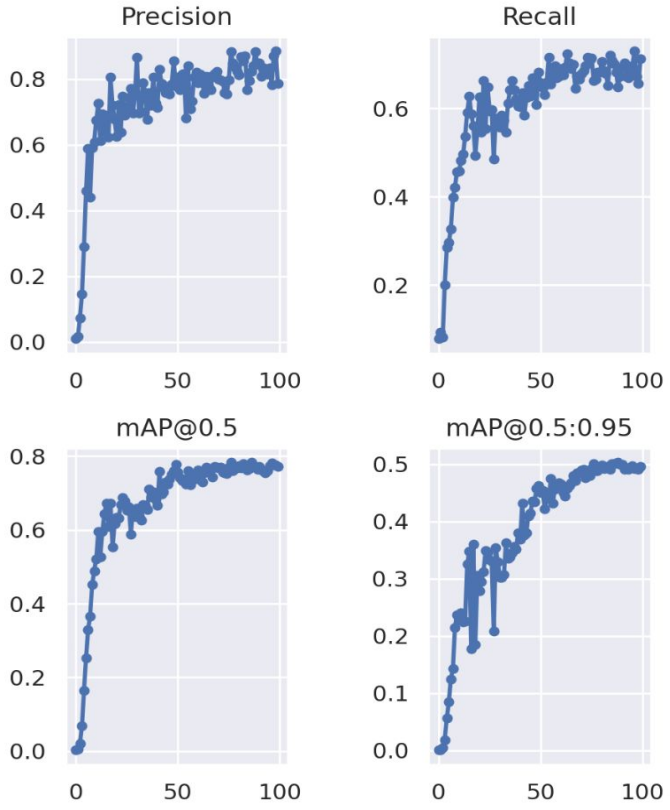
Ground truth

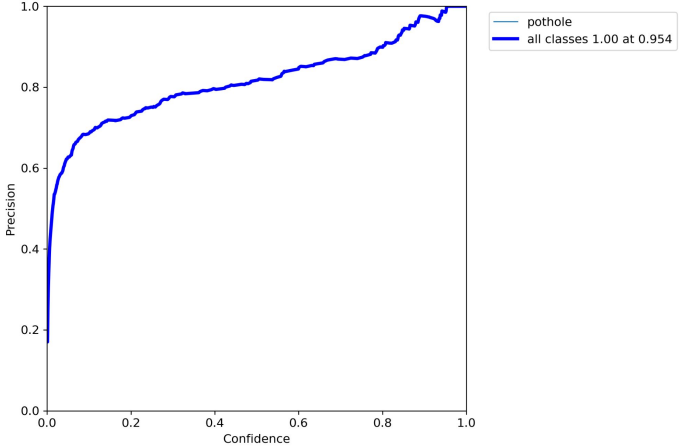
Predicted label

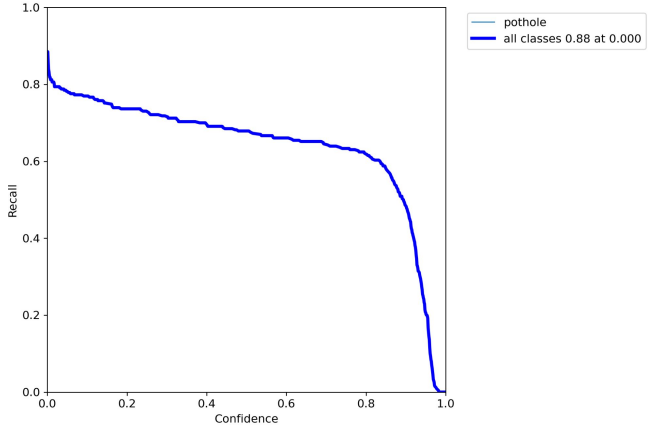
Inference on images

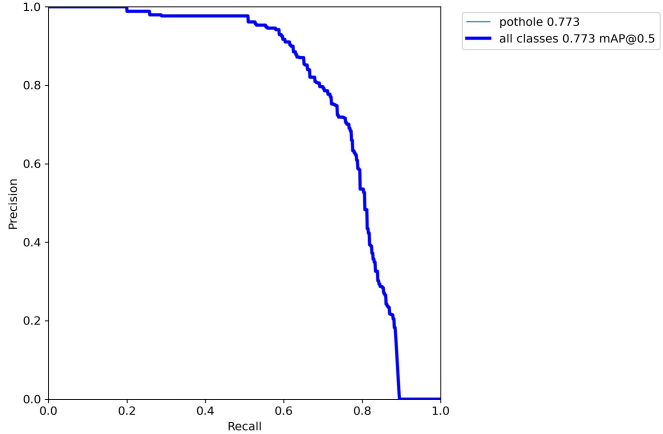


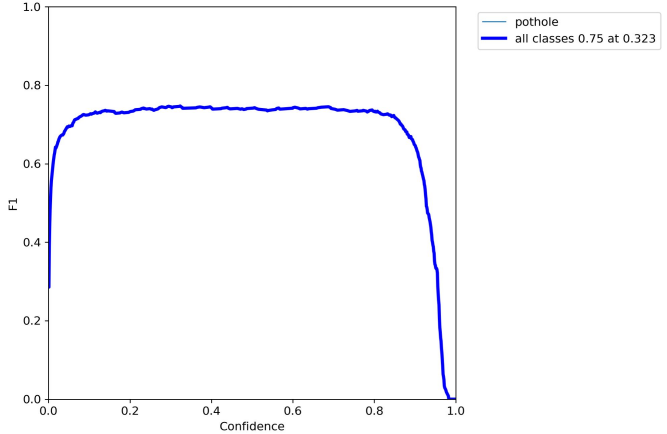
Confusion matrix




pothole
all classes 1.00 at 0.954
Precision
Confidence
0.0
0.2
0.4
0.6
0.8
1.0


pothole
all classes 0.88 at 0.000
Recall
Confidence
0.0
0.2
0.4
0.6
0.8
1.0


pothole 0.773
all classes 0.773 mAP@0.5
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0


pothole
all classes 0.75 at 0.323
F1
Confidence
0.0
0.2
0.4
0.6
0.8
1.0

Novel idea for future work.

IIT Hyderabad

- In foggy or rainy weather, performance of object detection models like YOLOv7 can significantly reduce.
- **Domain adaptation** can be a powerful technique to improve the performance of YOLOv7.
- A common technique is to use a combination of labeled data from the source domain (clear weather conditions) and unlabeled data from the target domain (foggy / rainy weather conditions).
- Self-training approach can be used, where the
  - Model is trained on the labeled data from the source domain.
  - Used to generate pseudo-labels for the unlabeled data from the target domain.
  - The model is then retrained on the combined labeled and pseudo-labeled data,

GOAL → Improve performance on target domain.

References

[1] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2016

[2] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2017

[3] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767, 2018.

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong- Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934, 2020.

[5] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. NAS-FPN: Learning scalable feature pyramid architecture for object detection. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019.

[6] Jocher Glenn. YOLOv5 release v6.1. https://github.com/ultralytics/yolov5/releases/tag/v6.1, 2022.

[7] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019.

[8] Chien-Yao Wang, Alexey Bochkovskiy, and Hong- Yuan Mark Liao. Scaled-YOLOv4: Scaling cross stage partial network. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2021.

[9] Shangliang Xu, Xinxin Wang, Wenyu Lv, Qinyao Chang, Cheng Cui, Kaipeng Deng, Guanzhong Wang, Qingqing Dang, Shengyu Wei, Yuning Du, et al. PPYOLOE: An evolved version of YOLO. arXiv preprint arXiv:2203.16250, 2022.

[10] Xianzhi Du, Barret Zoph, Wei-Chih Hung, and Tsung-Yi Lin. Simple training strategies and model scaling for object detection. arXiv preprint arXiv:2107.00057, 2021

IIT Hyderabad

Thank you!