Visual Object Tracking

Present By

Meet Patel - 400486035
Krish Patel - 400488152

OUTLINE

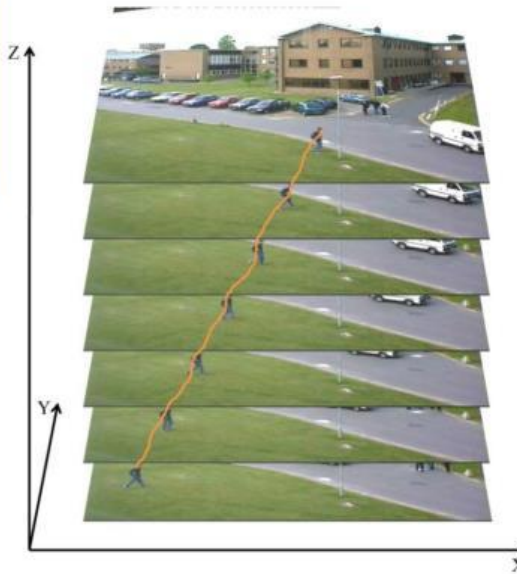
- PROBLEM STATEMENT
- DEFINITION
- BRIEF ABOUT PROJECT
- DATASET DESCRIPTION
- TASKS TO PERFORM THE PROJECT
- TOOLS AND TECHNOLOGIES TO BE USED
- TYPES OF OBJECT TRACKING
- DEEP LEARNING APPROACHES
- YOLO
- APPROACH OF OUR PROJECT
- PREPROCESSING
- YOLO V5
- YOLO V7
- CONCLUSION AND FUTURE APPROACH

PROBLEM STATEMENT

★ **Build intelligent system for performing the tracking of the everyday objects.**

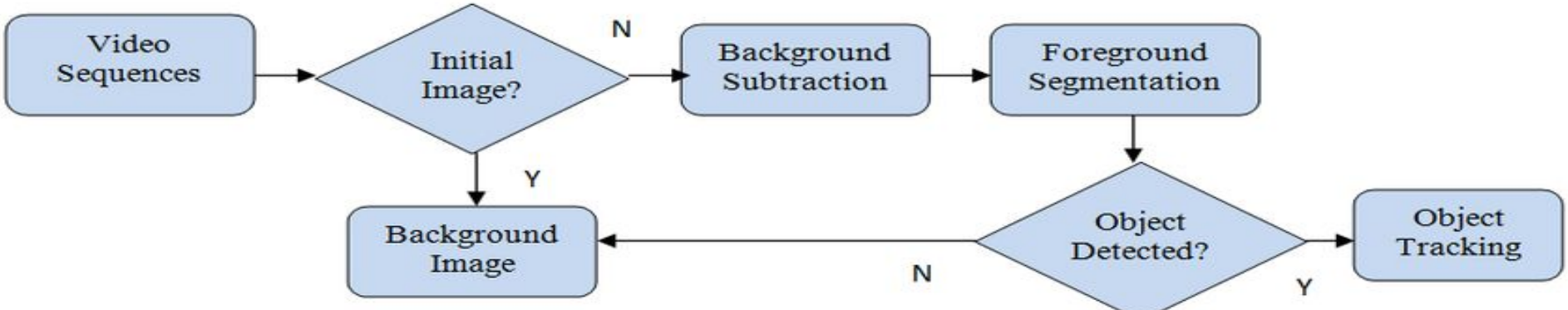
What is object tracking?

Estimating the trajectory of an object over time by locating its position in every frame.



DEFINITION

The video object tracking is to associate or establish a relationship between target objects as it appears in each video frame. In other words, video object tracking is analyzing the video frames sequentially and stitching the past location of the object with the present location by predicting and creating a bounding box around it. It is widely used in traffic monitoring, self-driving cars, and security because it can process real-time footage.



BRIEF ABOUT PROJECT

Overall, In this project, we will focus on the object tracking into the field of the view, where the goal is to develop a system and deep learning model that can accurately track objects from the given input image.

Object tracking refers to the ability of a computer system or software application to keep track of objects within its viewing area, and to update the display of those objects as they move or change.

Among the many different applications of computer vision, object tracking is one of the most important components. such as, real world objects, autonomous driving, human-computer interaction and facial detection.

DATASET DESCRIPTION

The cvlab under the hanyang university of the south korea created the visual tracker benchmark dataset for the computer vision experiments. It contains 100 classes of the different real world objects and every classes have the sequences of the images which was captured from the single video.

Dataset name	Visual Tracker Benchmark
Dataset link	http://cvlab.hanyang.ac.kr/tracker_benchmark/datasets.html
Dataset Type	Images
Number of Class	100 sequences of the class
Number of image per sequence	250 - 2000
Size of each image	Up to 60 kb

TASKS TO PERFORM THE PROJECT

Task 1: Data Preprocessing

Task 2: Target initialization

Task 3: Appearance modeling

Task 4: Motion estimation

Task 5: Target positioning

TOOLS AND TECHNOLOGIES USED

TOOLS

- Roboflow
- Google Colab

TECHNOLOGIES

- Python
- Deep Learning
- Torch
- YOLOv5
- YOLOv7

TYPES OF OBJECT TRACKING

1. **Single Object Tracking(SOT)**
2. **Multiple Object Tracking(MOT)**

Here, as we are dealing with the visual object tracking, we are performing Single Object Tracking(SOT) rather than Multiple Object Tracking(MOT).

The basic goal of SOT system is to track a single object of interest over long periods of time under varying conditions.

SOT belongs to the category of detection-free tracking because one has to manually provide the first bounding box to the tracker. This means that Single Object Trackers should be able to track whatever object they are given, even an object on which no available classification model was trained.

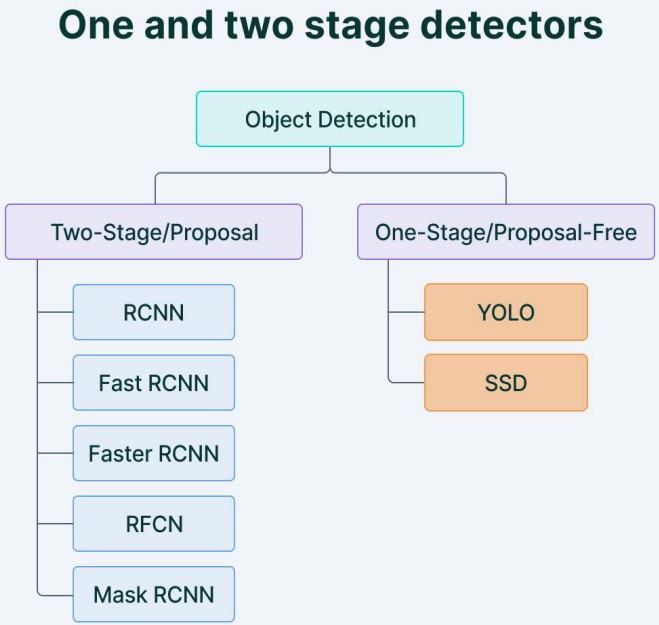
DEEP LEARNING APPROACHES

- We are using deep learning approach rather than traditional ML as it extract important features and representations by themselves.

- Till date, a lot of methods and ideas were introduced to improve the accuracy and efficiency of the tracking models. Including CNN, RCNN, RFCN, LSTM, YOLO.

- Fast RCNN, Faster RCNN, RFCN are two-step algorithms so they are break down the object detection problem statement into the following two-stages:

1. Detecting possible object regions.
2. Classifying the image in those regions into object classes



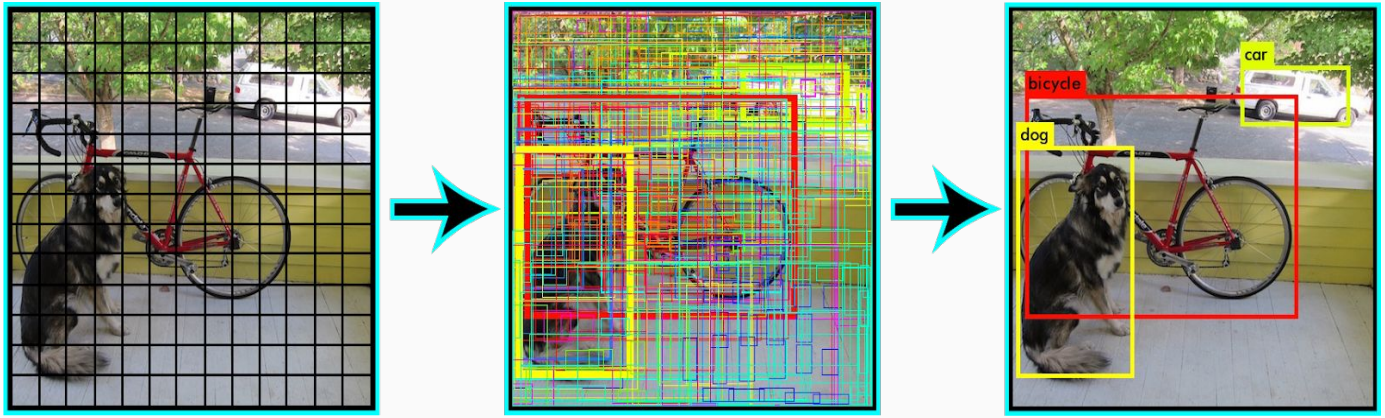
- In the case of high mean Average Precision (mAP), it results in multiple iterations taking place in the same image, thus slowing down the detection speed of the algorithm and preventing real-time detection.

- Moreover, to increased accuracy in predictions and a better Intersection over Union in bounding boxes (compared to real-time object detectors), YOLO has the inherent advantage of speed. It is also much faster algorithm than its counterparts, running at as high as 91 FPS.

- Hence, at the end we have chosen the YOLO approach for our entire project.

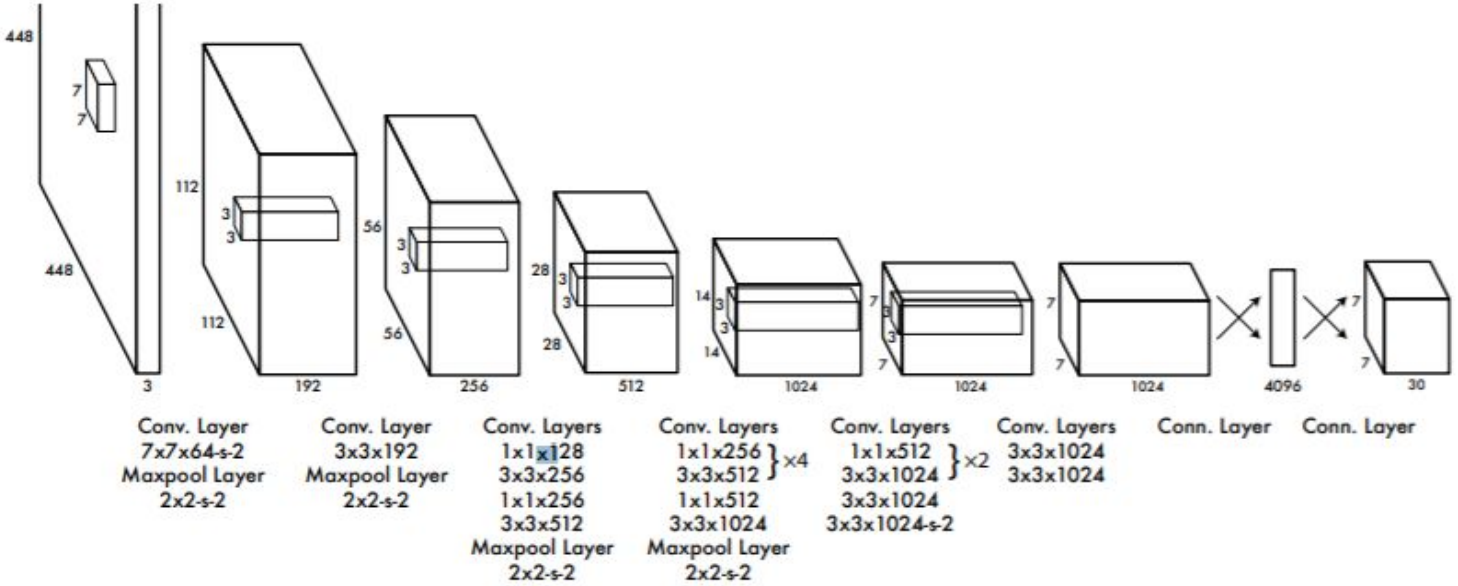
YOLO

The YOLO algorithm works by dividing the image into *N* grids, each having an equal dimensional region of SxS. Each of these *N* grids is responsible for the detection and localization of the object it contains.



- **YOLO Architecture**

Motivated by the GoogleNet architecture, YOLO's architecture has a total of 24 convolutional layers with 2 fully connected layers at the end.



APPROACH OF OUR PROJECT

- When we opted to use the YOLO technique and downloaded the dataset, we learned that the dataset was not intended for the YOLO algorithm as the labeling was not done in the format of Yolo
- Therefore, we decide to annotate the dataset using roboflow to convert it into the yolov5 and yolov7 annotation formats which are the two algorithm we used for object tracking.
- We employed four different dataset classes: Biker, Blurcar, Bird1, and Person.

PREPROCESSING

Roboflow also offers a variety of preprocessing processes from which we decided to employ following steps :

- Auto orientation
- Resizing - 640*640
- Converting to grayscale

After doing above steps, in order to improve the results, we used augmentations in which we generated 3x training images and flipped the images into horizontal and vertical orientations.

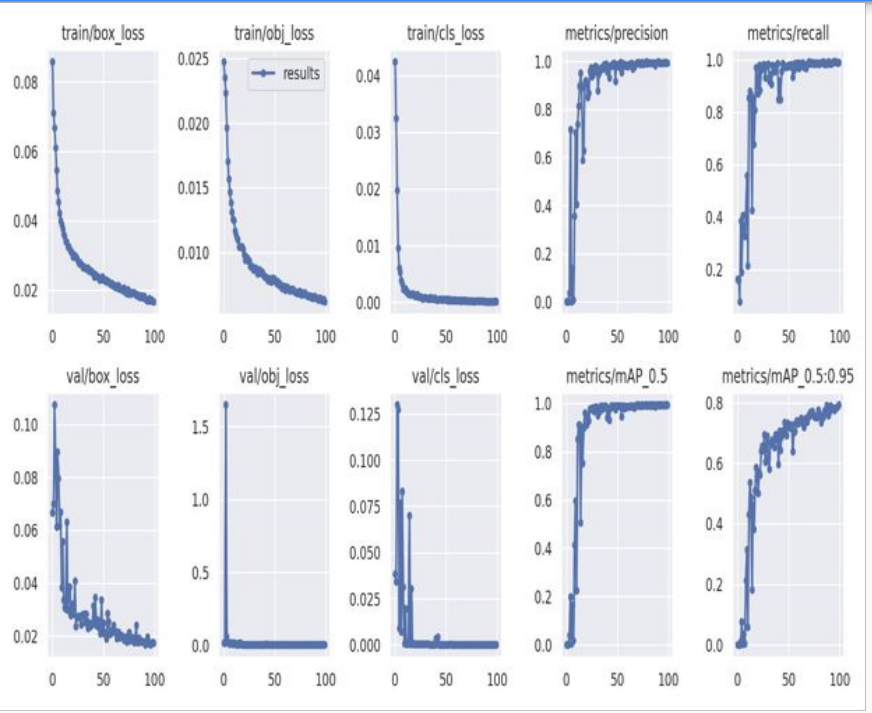
After doing all this steps we have

- Training set : 2.7k images (87.4%)
- Validation set : 230 images (7.3%)
- Test set : 165 images (5.3%)

YOLO V5

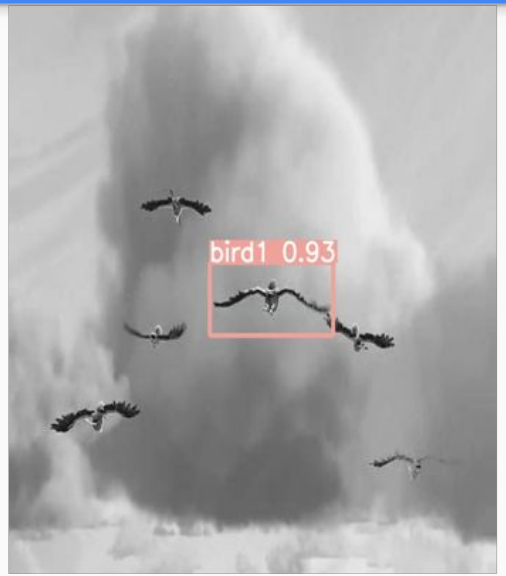
- There are other YOLO V5 variations, but the four main ones are Small, Medium, Large, and Extra Large. Depending on which one you choose, the accuracy rates and parameter values will vary. However, it will also take more time and be more expensive computationally.
- Here we have used small version as it would save us time and give us a relatively good accuracy.
- We ran it for 100 epochs with batch size of 64 images.
- We used best parameter from training yolov5 to run on test images for results.

YOLO V5 RESULTS

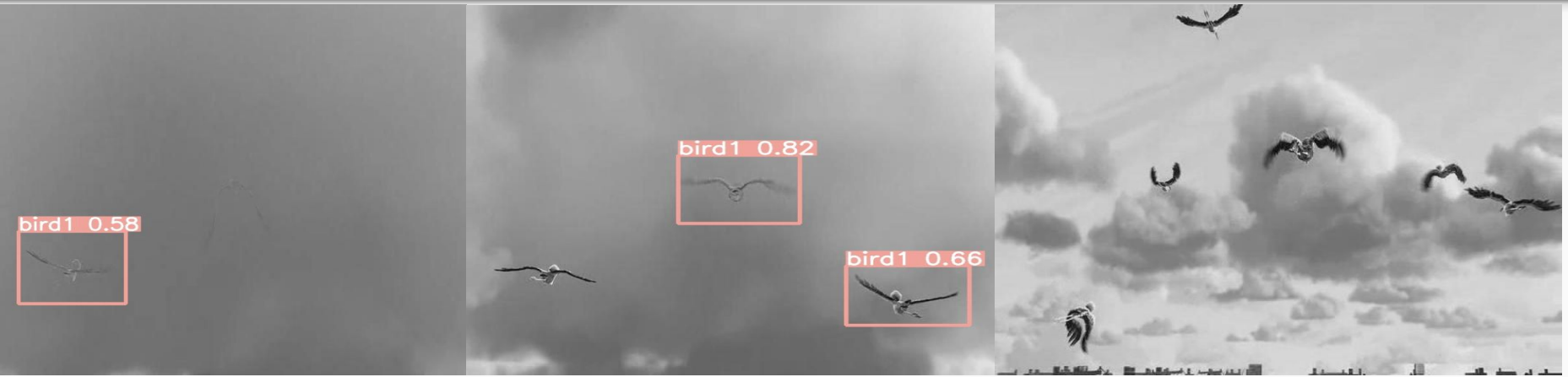



```
     Epoch   gpu_mem       box       obj       cls    labels  img_size
     98/99       13G   0.01695  0.006385 0.0002599       116       640: 100% 42/42 [00:34<00:00,  1.22it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 2/2 [00:02<00:00,  1.32s/it]
                  all        230        219      0.996      0.989      0.993      0.788

     Epoch   gpu_mem       box       obj       cls    labels  img_size
     99/99       13G   0.01669  0.006186 0.0001206       118       640: 100% 42/42 [00:33<00:00,  1.24it/s]
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 2/2 [00:02<00:00,  1.34s/it]
                  all        230        219      0.993      0.991      0.994      0.793

100 epochs completed in 1.018 hours.
Optimizer stripped from runs/train/yolov5s_results/weights/last.pt, 14.5MB
Optimizer stripped from runs/train/yolov5s_results/weights/best.pt, 14.5MB

Validating runs/train/yolov5s_results/weights/best.pt...
Fusing layers...
YOLOv5s summary: 213 layers, 7020913 parameters, 0 gradients, 15.8 GFLOPs
                Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 2/2 [00:04<00:00,  2.06s/it]
                  all        230        219      0.993      0.991      0.994      0.793
                Biker        230         14      0.996          1      0.995      0.777
                bird1        230         71      0.986      0.965      0.991      0.723
              blurcar        230        117      0.998          1      0.995      0.849
               person        230         17      0.994          1      0.995      0.825
Results saved to runs/train/yolov5s_results
CPU times: user 30 s, sys: 3.82 s, total: 33.8 s
Wall time: 1h 2min 21s
```

YOLO V5 ON TEST IMAGES



YOLO V5 ON TEST IMAGES



Here when our algorithm detected on test set we can see that when detecting bird1 there were inconsistency but when detecting others it did a pretty good job.

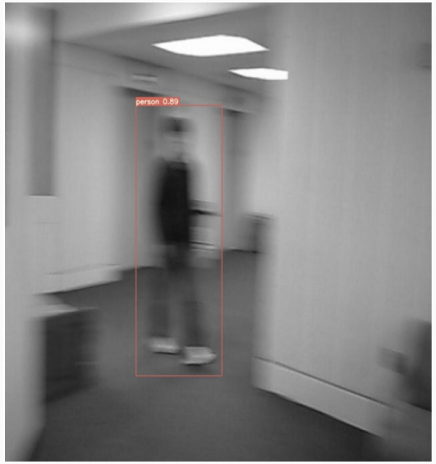
YOLO V7 RESULTS

Yolov7 contains 72.12 Million parameters and we ran it for 100 epochs with batch size of 16 in google colab and here are the results we got:

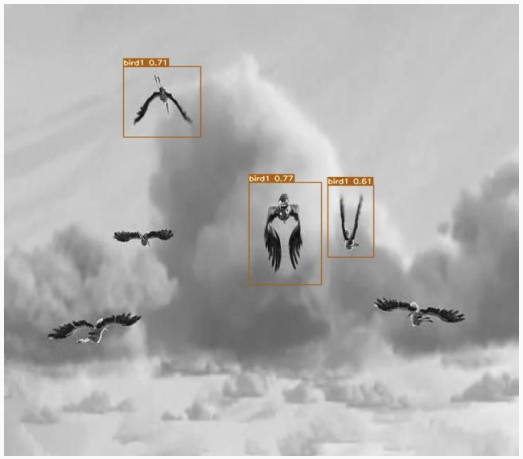
```
     Epoch    gpu_mem       box       obj       cls     total    labels  img_size
     96/99      10.9G   0.02454  0.003734 0.0008942   0.02917        18       640: 100% 167/167 [02:19<00:00,  1.19it/s]
                Class      Images      Labels         P         R      mAP@.5  mAP@.5:.95: 100% 8/8 [00:04<00:00,  2.00it/s]
                  all         230         219     0.975     0.972       0.99       0.767

     Epoch    gpu_mem       box       obj       cls     total    labels  img_size
     97/99      10.9G   0.02435  0.003625 0.0008991   0.02887        13       640: 100% 167/167 [02:19<00:00,  1.20it/s]
                Class      Images      Labels         P         R      mAP@.5  mAP@.5:.95: 100% 8/8 [00:03<00:00,  2.01it/s]
                  all         230         219      0.98     0.962       0.986       0.768

     Epoch    gpu_mem       box       obj       cls     total    labels  img_size
     98/99      10.9G   0.02454  0.003526 0.0008303    0.0289        19       640: 100% 167/167 [02:17<00:00,  1.21it/s]
                Class      Images      Labels         P         R      mAP@.5  mAP@.5:.95: 100% 8/8 [00:03<00:00,  2.01it/s]
                  all         230         219     0.972     0.986       0.99       0.746

     Epoch    gpu_mem       box       obj       cls     total    labels  img_size
     99/99      10.9G   0.02369  0.003667 0.0009108   0.02827        12       640: 100% 167/167 [02:19<00:00,  1.20it/s]
                Class      Images      Labels         P         R      mAP@.5  mAP@.5:.95: 100% 8/8 [00:05<00:00,  1.48it/s]
                  all         230         219     0.962     0.982       0.988       0.756
                Biker         230          14     0.997         1       0.995        0.74
                bird1         230          71     0.892     0.929       0.965       0.666
              blurcar         230         117     0.989         1       0.995       0.827
               person         230          17     0.972         1       0.995       0.791
100 epochs completed in 4.017 hours.
```

YOLO V7 ON TEST IMAGES

The best parameters from training yolov7 were used to get the results on test images.



YOLO V7 ON TEST IMAGES



Here similarly as yolov5 algorithm, yolov7 also has inconsistency in detecting bird1 but does a good job in detecting others.

CONCLUSION AND FUTURE APPROACH

Here, we may infer that our model is performing a respectable job of object detection even on test photos, which are data that it has never seen before. Although there are some issues with the bird class detection, the other three classes are handled very well. In this project, we used the provided dataset to perform SOT Visual Object Tracking. We chose the Deep Learning Approach - YOLO for attaining its application rather than regular ML because it has several advantages that have previously been stated in this research. Right present, we are using two YOLO models in the project: YOLO5 and YOLO7.

To enhance our model, we can build a larger dataset with more classes and use it for multiple object tracking in a single image. This will allow our model to be trained on a larger volume of data, which will help improve our model's accuracy and we can detect the object more efficiently.


THANK YOU