DEEP LEARNING - SEP 740 - FALL 2022

ENGINEERING
W Booth School of Engineering
Practice and Technology

McMaster University

COURSE PROJECT

Final Project Report

Project 9: Visual Object Tracking

Submitted By

Meet Patel - 400486035 ✉***patem207@mcmaster.ca***

Krish Patel - 400488152 ✉***patek179@mcmaster.ca***

Master Of Engineering System & Technology

Automation & Smart System

`

★ ABSTRACTION - THE PROBLEM STATEMENT

Build visual object tracking system for performing the detection of the everyday objects.

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, human-computer interaction and facial detection.

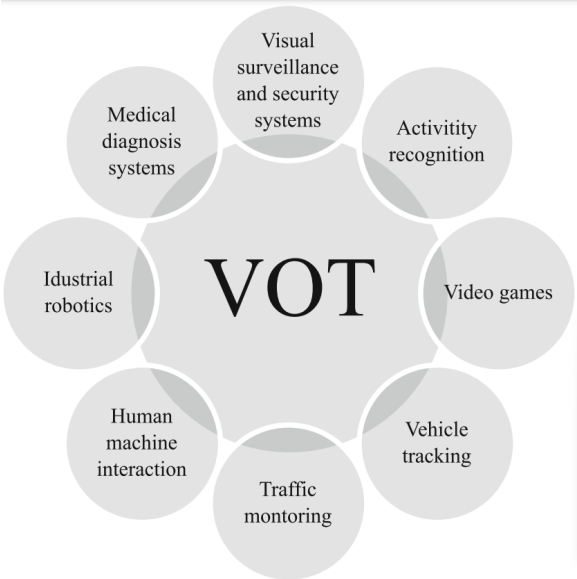


Fig. 1 Usage of VOT in different applications

★ DATASET DESCRIPTION

The cvlab under the hanyang university of the south korea was 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. We are building the system from the some certain sequences of the top 50 sequences.

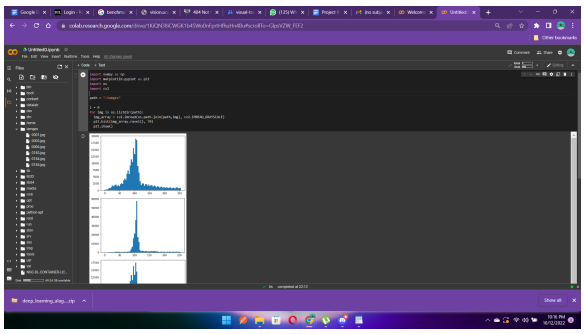
Dataset name	Visual Tracker Benchmark
Dataset link	http://cvlab.hanyang.ac.kr/tracker_benchmark/datasets.html
Dataset Type	Images
Number of Class	100 sequences of the class
Number of image per sequence	250 - 2000
Size of each image	Up to 60 kb

The attribute distribution of the data set is represented In the followingt table. Every sequence of the whole database is tagged with the multiple attributes. As per the below table, the more common ones are OPR and IPR.which are tagged frequently

★ DATA VISUALISATION

We visualized some frames of the basket ball sequences from the given dataset. The histogram displays the number of pixels on the vertical axis with the particular tonal value on the horizontal axis as well as it is showing the continuous data without any gap between it. Hence as per our knowledge, histogram is the convenient plot to represent our data.

We have attached the screenshot of data visualisation and histogram of some images.



Steps we will follow to achieve our goal:

Stage 1:

For the first step of the object tracking process, we will analayzed sequences o the video frames to recognize individual objects such as people, animals, vehicles, tree or other objects in the scene. The image from each frame will be analyzed to determine the position and orientation of each object.

Stage 2:

we will track the images using an object detection and tracking algorithm. During the second stage of the process, the algorithm attempts to track the position of each object in each frame of the video. This process usually involves comparing an image in frame to images already processed in order to find similar features in the images.

Stage 3:

We will try to build a algorithms which can segment the video sequence according to a certain period of time, and detect each object individually according to the set of characteristics found in it. For example,

each object is characterized with a certain color and shape depending on the object type.

Stage 4:

Then, based on the previous steps, our algorithm will identify the location of the object in each frame and make a record of its position. The position of an object will be tracked by estimating its bounding box.

A bounding box is a region around an object that is calculated using certain properties of the object such as color, shape and texture. This box will be use to accurately determine the location and movement of the object over time. Then, we will store this data for further analysis.

★ TASK DESCRIPTION

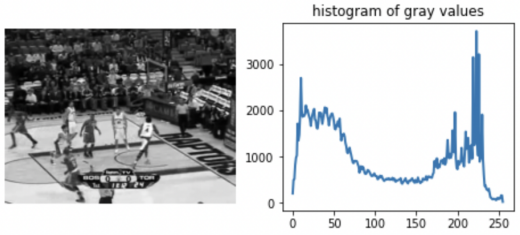
First significant difficulty with object tracking is not only to classify the object but also to establish the location of the object as to where it is present in the image.

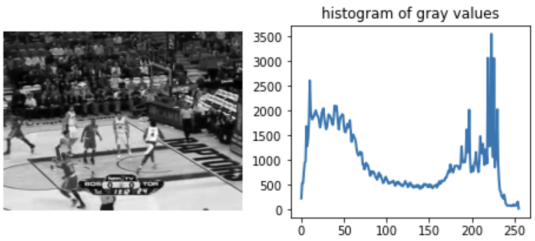
Based on the fact that the goal of our project is to visualize various objects, this task involves both classification and regression because we must accurately classify and detect various objects in our dataset as well as continuously detect them in the video, which also constitutes a regression task.

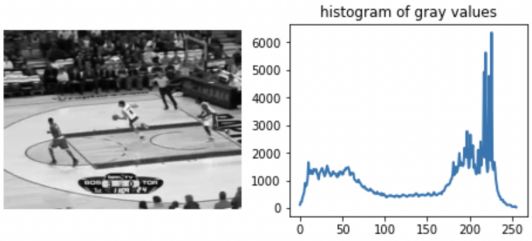
We will be using classification for prediction of different objects or different kind of objects whereas regression task will be used to predict the location of bounding box use to predict the object.

Hence our task includes both classification and regression problem which we think will be useful to improve final prediction with much more accuracy.

- **Distribution of the data using histogram is as follows :**







From our dataset we used basketball dataset for our segmentation and detection

First we used a random image from basketball dataset to train a machine learning model and then that model will be used for all images in dataset

Now we convert RGB images into grayscale images for segmentation. Here we have used different filters like Gabor, Gaussian, Scharr, Sobel etc for feature extraction. Then we split the data into 80%,20% for training and testing dataset respectively . Then we used 2 different types of ml models i.e. random forest classifier and support vector machine for predicting the output. We used 10 estimators for random forest classifier and 100 iterations for SVM, it took a long long time to converge in SVM when we used 5000 iterations.

Then we test the accuracy on train and test data which is as follows for

Random forest classifier

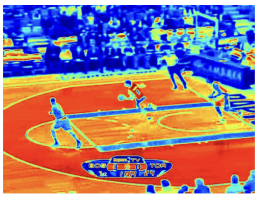
```
Accuracy on training data =  0.9989752091025544
Accuracy =  0.8499005365000905
```

Support vector machine

```
Accuracy on training data =  0.021575867179062114
Accuracy =  0.020374947254204592
```

Hence we used random forest classifier for our for our prediction and then saved the model using pickle and then used that model to segment all images in the dataset

Our original images and segmented images are as follows -



★ TYPES OF THE 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.

★ REASON TO CHOOSE DL INSTEAD OF TRADITIONAL ML -

As we are dealing with computer vision problem, DL is the best method to reach out to the best approach. DL-based models can learn complex representations better and faster than other ML techniques. In addition, DL requires fewer labeled data samples and can also outperform existing ML methods when there is an imbalance in classes

As we have large dataset, DL models are able to achieve good performance for a wide range of tasks including image classification and object detection. Our these applications typically involve processing large amounts of data that cannot be handled efficiently by traditional ML techniques.

The availability of powerful compute resources and large datasets have enabled the development of sophisticated DL algorithms that are capable of learning good representations of data and outperforming all other ML methods in many application domains.

Many DL algorithms have the potential to be trained on only a few labeled samples and perform accurately on larger unseen datasets without the need for additional manual feature engineering or tuning.

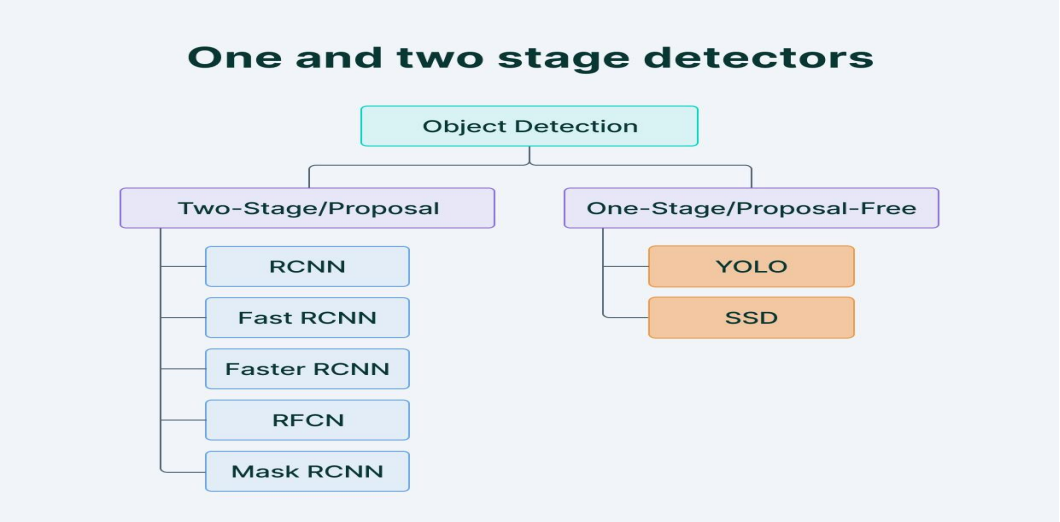
★ **DEEP LEARNING APPROACHES** -

- 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.

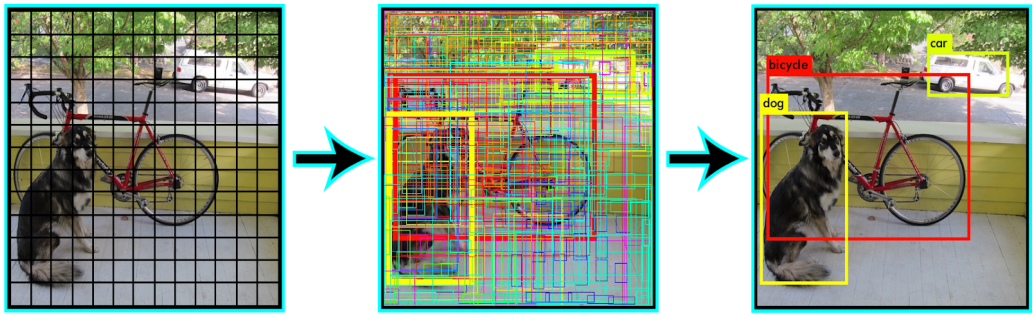
- Hence, at the end we have chosen the YOLO approach for our entire project.

- **After exploring and evaluating deep learning models, we have realised that YOLO will performed well on our dataset.**

.

★YOLO :

This is a well-known deep learning technique that was created recently and stands for "you only look once." The architecture of this algorithm basically consist of different Convolution layers to detect object. This algorithm's popularity is primarily due to the fact that it forecasts the bounding box and class probabilities in only one run, something that other algorithms don't do. As a result, it makes predictions significantly faster than other detection algorithms that are currently in use, with high accuracy and great learning capabilities. It basically uses 3 techniques for detection and that is non max suppression, bounding box regression and residual blocks.



- Residual blocks: it just divides the input image into n*n grids and now every grid will detect the objects within themselves

- Bounding box regression: The characteristics present in each bounding box in the image are its height, width, class and centre of bounding box

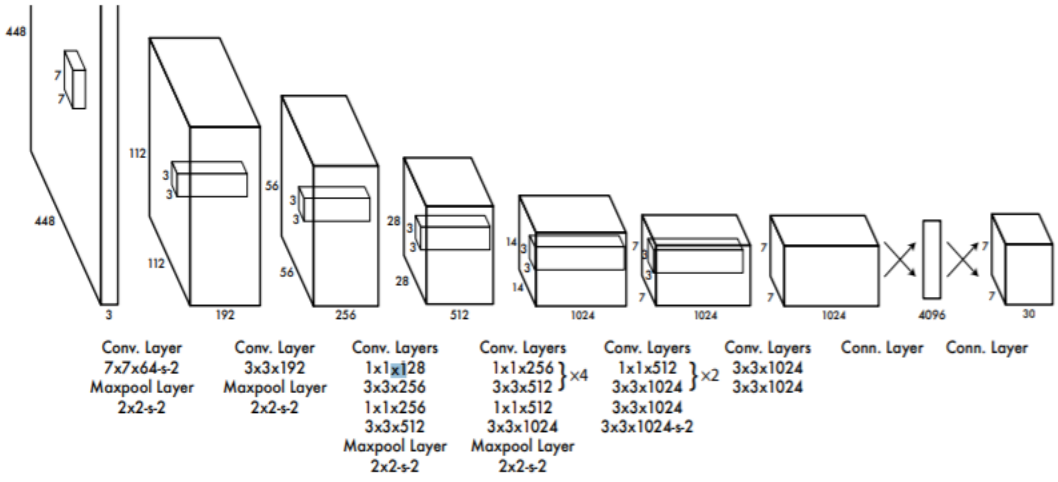
- non max suppression: now for this IOU is performed with respect to the bounding box with the highest class probability and excludes the bounding boxes which has higherIOU value than a certain threshold.

After combining all these 3 techniques we can detect different objects in given input image or video or live feed. It is computationally less expensive and recognises the object more accurately and quickly than any machine learning models.

There have been different Yolo models nowadays for our use which are YOLO V2, YOLOV3, YOLOV4, YOLOV5 and YOLOV7.

- **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 THE 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.

To transfer all images into our model, we resized them into a single dimension. To do this, we performed preprocessing and scaling each image's height and width to the equal size.

Here, No need to do the **missing values** as all images are in the sequence and in the same name format. As well as we are approaching deep learning models to perform the task, we **no need to do the manual feature extraction**. It will learn and extract automatically while doing training the models.

★ DATA PREPROCESSING

In our project, as it includes image dataset we dont require to do much pre processing.

Converting color images into grey scale image -

Since our task is visual object tracking, we first transformed an RGB image into a grayscale image for preprocessing for our detection task. Color is not really necessary and by doing so we can reduce complexity and memory usage.

Standardize image -

We used Roboflow which 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%)

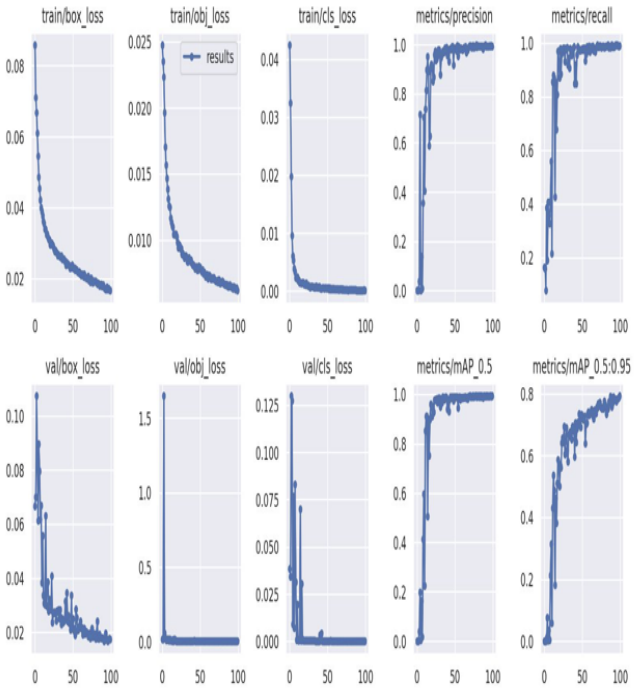
- Validation set : 230 images (7.3%)
- Test set : 165 images (5.3%

★ RESULTS

YOLOV5

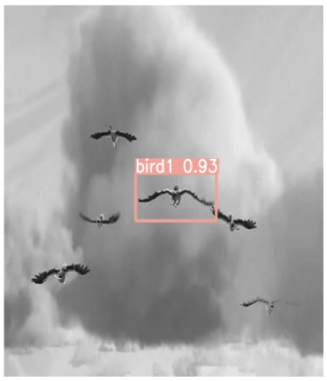
- There are other YOLO V5 variations, but the four main ones are Small, Medium, Large, and Extra Large. Depending on which one you choose, the accuracy rates and parameter values will vary. However, it will also take more time and be more expensive computationally.
- Here we have used small version as 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.

YOLOV5 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
```

YOLOV5 ON TEST IMAGES

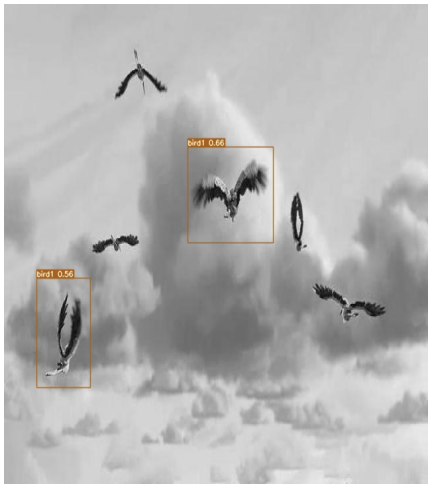


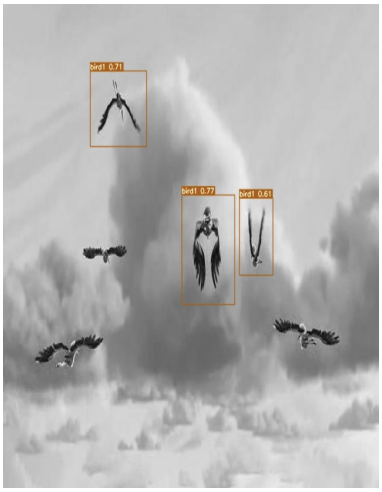
YOLOV7

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.
```

YOLOV7 ON TEST IMAGES





★ CONCLUSION

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.

We are using two YOLO models in this project: YOLO5 and YOLO7. Here, we can infer that our models are performing a respectable job of object detection even on test images, 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 and bird class is also handled not that badly. But we can also state that Yolov5 performs well in a lot less time than Yolov7, as Yolov5 took only approximately one hour compared to Yolov7's four hours, and Yolov7 also has a little bit less mAP than Yolov5. We can therefore draw the conclusion that YOLOV5 performed better than YOLOV7 for our dataset and was also computationally less expensive.

★ LIMITATIONS OF THE MODELS

When we ran our model on our dataset of 4 classes it did a good job in classifying 3 classes which are car, person, biker but in detecting bird1 it created some problems like detecting more than one bird or not detecting any birds or detecting the wrong bird on both yolov5 and yolov7. Also it was trained for single object tracking with just 4 classes with much less data.

★ FUTURE WORKS

To enhance our model, we can build a larger dataset with more classes and use it for multiple object tracking. 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. To obtain a model that is more accurate, we can also employ additional yolov5 models, such as medium, large, or others

★ GITHUB REPOSITORY

https://github.com/kp0402/demopygit