GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

MULTIPLE OBJECT DETECTION SYSTEM

A Project Report of Capstone Project - 2

Submitted by

DHRUV JAIN

(1613101258)

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of

Mr. MUKESH KUMAR JHA,

Assistant Professor

APRIL / MAY- 2020

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report **"MULTIPLE OBJECT DETECTION SYSTEM"** is the bonafide work of **"DHRUV JAIN(1613101258)"** who carried out the project work under my supervision.

SIGNATURE OF HEAD

Dr. MUNISH SHABARWAL,
PhD (Management), PhD (CS)
Professor & Dean,
School of Computing Science & Engineering

SIGNATURE OF SUPERVISOR

Mr. MUKESH KUMAR JHA,
Assistant Professor
School of Computing Science & Engineering

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Object detection is one of the major aspect in computer environment. It is highly used for detecting faces, vehicle, pedestrian, security, self-driving cars, retina scan, brain scan and many more. We are using highly accurate object detection-algorithms and methods like R-CNN, Faster-RCNN, COCO model and YOLO (You Only Look Once). We are using several packages in this process like TensorFlow, OpenCV, Darknet, CUDA. We can recognize and detect each and every object in an image with the help of the region highlighted as rectangular boxes and assign a name tag to the object. In object detection dataset plays a very crucial role as we have to train data i.e in object detection we perform two operations on the data that are training data and testing data. So in the training phase we see what type of object is there in the data and then we highlight the data with the help of rectangular boxes. After that we test whether we have trained the data accurately or not. Object detection is not only implied on images but we can also use it for detecting objects in a video. One of the most burning example in todays world is face lock in smartphones. It also uses the same concept of detection. It detects the retina and then work accordingly on the face lock.

1. INTRODUCTION

1.1. Overall Description

Object recognition is to describe a collection of related computer vision tasks that involve activities like identifying objects in digital photographs. Image classification involves activities such as predicting the class of one object in an image. Object localization is refers to identifying the location of one or more objects in an image and drawing an abounding box around their extent. Object detection does the work of combines these two tasks and localizes and classifies one or more objects in an image. When a user or practitioner refers to the term "object recognition", they often mean "object detection". It may be challenging for beginners to distinguish between different related computer vision tasks.

So, we can distinguish between these three computer vision tasks with this example:

Image Classification: This is done by Predict the type or class of an object in an image.

Input: An image which consists of a single object, such as a photograph.

Output: A class label (e.g. one or more integers that are mapped to class labels).

Object Localization: This is done through, Locate the presence of objects in an image and indicate their location with a bounding box.

Input: An image which consists of one or more objects, such as a photograph.

Output: One or more bounding boxes (e.g. defined by a point, width, and height).

Object Detection: This is done through, Locate the presence of objects with a bounding box and types or classes of the located objects in an image.

Input: An image which consists of one or more objects, such as a photograph.

Output: One or more bounding boxes (e.g. defined by a point, width, and height),

and a class label for each bounding box.

One of the further extension to this breakdown of computer vision tasks is object segmentation, also called “object instance segmentation” or “semantic segmentation,” where instances of recognized objects are indicated by highlighting the specific pixels of the object instead of a coarse bounding box.
For example, image classification is simply straight forward, but the differences between object localization and object detection can be confusing, especially when all three tasks may be just as equally referred to as object recognition.
Humans can detect and identify objects present in an image. The human visual system is fast and accurate and can also perform complex tasks like identifying multiple objects and detect obstacles with little conscious thought. The availability of large sets of data, faster GPUs, and better algorithms, we can now easily train computers to detect and classify multiple objects within an image with high accuracy. We need to understand terms such as object detection, object localization, loss function for object detection and localization, and finally explore an object detection algorithm known as “You only look once” (YOLO) and also on COCO model.

Object recognition refers to a collection of related tasks for identifying objects in digital photographs. Region-based Convolutional Neural Networks, or R-CNNs, is a family of techniques for addressing object localization and recognition tasks, designed for model performance. You Only Look Once, or YOLO is known as the second family of techniques for object recognition designed for speed and real-time use. Firstly we have used a COCO model data set i.e. Common Object in Context, by Microsoft. It is a large-scale object detection, segmentation, and captioning dataset.

1.2. Purpose

The aim of object detection is to detect all instances of objects from a known class, such as people, cars or faces in an image. Generally, only a small number of instances of the object are present in the image, but there is a very large number of possible locations and scales at which they can occur and that need to somehow be explored. Each detection of the image is reported with some form of pose information. This is as simple as the location of the object, a location and scale, or the extent of the object defined in terms of a bounding box. For example for face detection in a face detector may compute the locations of the eyes, nose and mouth, in addition to the bounding box of the face. An example of a bicycle detection in an image that specifies the locations of certain parts is shown in Fig. 1. The pose can also be defined by a three-dimensional transformation specifying the location of the object relative to the camera. Object detection systems always construct a model for an object class from a set of training examples. In the case of a fixed rigid object in an image, only one example may be needed, but more generally multiple training examples are necessary to capture certain aspects of class variability.

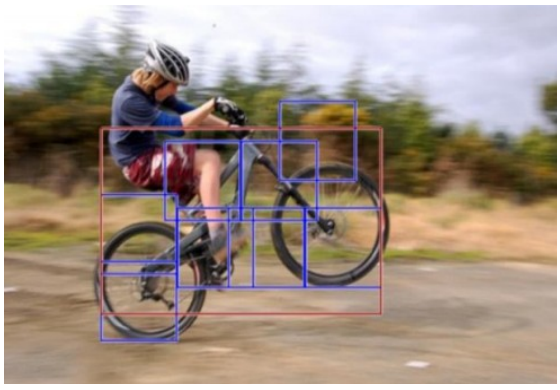

Fig.1 Bicycle with Bounding Boxes

1.3. Hardware Requirements

- Processor - i3
- RAM - 4 GB
- Memory – 5 GB

1.4. Software and Library Requirements

- Python 3.7.2
- Google colab
- TensorFlow
- CUDA
- Darknet
- OpenCV
- COCO Dataset & YOLO Weights

1.5. Applications and Future Scope

1.5.1. Optical Character Recognition

Optical character recognition or optical character reader, often abbreviated as OCR, is the mechanical or electronic conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image, we are extracting characters from the image or video.

Fig. 2 OCR

1.5.2. Self-Driving Cars

One of the best examples of why you need object detection is for autonomous driving is In order for a car to decide what to do in next step whether accelerate, apply brakes or turn, it needs to know where all the objects are around the car and what those objects are That requires object detection and we would essentially train the car to detect known set of objects such as cars, pedestrians, traffic lights, road signs, bicycles,motorcycles, etc.

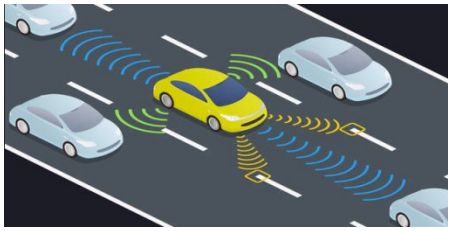

Fig. 3 Self-Driving Cars

1.5.3. Tracking Objects

Object detection system is also used in tracking the objects, for example tracking a ball during a football match, tracking movement of a cricket bat, tracking a person in a video. Object tracking has a variety of uses, some of which are surveillance and security, traffic monitoring, video communication, robot vision and animation.

Fig. 4 Tracking Objects

1.5.4. Digital Watermarking

A digital watermark is a kind of marker covertly embedded in a noise-tolerant signal such as audio, video or image data. It is typically used to identify ownership of the copyright of such signal. "Watermarking" is the process of hiding digital information in a carrier signal; the hidden information should, but does not need to, contain a relation to the carrier signal. Digital watermarking may be used for a wide range of applications such as Copyright protection, Source tracking (different recipients get differently watermarked content), Broadcast monitoring (television news often contains watermarked video from international agencies), Video authentication, Software crippling on screencasting and video editing software programs, ID card security, Fraud and Tamper detection, Content management on social networks.

1.5.5. Medical Imaging

Medical image processing tools are playing an increasingly important role in assisting the clinicians in diagnosis, therapy planning and image-guided interventions. Accurate, robust and fast tracking of deformable anatomical objects such as the heart, is a crucial task in medical image analysis.

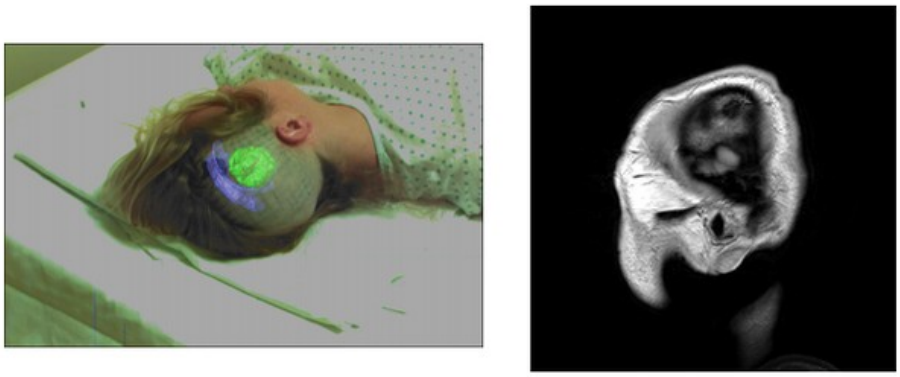

Fig. 5 Medical Imaging

1.5.6. Automated CCTV

Surveillance is an integral part of security and patrol. Recent advances in computer vision technology have lead to the development of various automatic surveillance systems, however their effectiveness is adversely affected by many factors and they are not completely reliable. This study investigated the potential of automated surveillance system to reduce the CCTV operator workload in both detection and tracking activities.

Normally CCTV is Running every time, so we need large size of memory system to store the recorded video. By using object detection system we can automate CCTV in such a way that if some objects are detected then only recording is going to start. Using this we can decrease the repeatedly recording same image frames, which increases the memory efficiency. We can decrease the memory requirement by using this object detection system.

2. LITERATURE REVIEW

2.1. Object Detection

Object detection is an important task, yet challenging vision task. It is a critical part of many applications such as image search, image auto-annotation and scene understanding, object tracking. Moving object tracking of video image sequences was one of the most important subjects in computer vision. It had already been applied in many computer vision fields, such as smart video surveillance (Arun Hampapur 2005), artificial intelligence, military guidance, safety detection and robot navigation, medical and biological application. In recent years, a number of successful single-object tracking system appeared, but in the presence of several objects, object detection becomes difficult and when objects are fully or partially occluded, they are obtruded from the human vision which further increases the problem of detection. Decreasing illumination and acquisition angle. The proposed

MLP based object tracking system is made robust by an optimum selection of unique features and also by implementing the Adaboost strong classification method.

2.2. Preprocessing

As the name suggest a level before processing. It is same as editing a photo. Before we post our photo we apply effects on that so in the same way preprocessing works. It improves the quality of the image by reducing noise and unwanted features and enhancing those features that are important for that image. It also resizes an image and make it to 448*448 size thereby managing the contrast and brightness of the image. It makes the image ready for feature extraction there by normalizing it for better fitting. In mathematical terms preprocessing can be achieved by removing or subtracting the mean value of image intensity and dividing it by standard deviation.

2.3. Feature Extraction

As the name suggests extracting the features. It is same as data mining. As in data mining we take only that data which is important for us means with which we can solve most problems. In the same way feature extraction works , in this we extract those features of the image which are important for our process there by ignoring rest of the features. One of the technique used for feature extraction is edge detection. Feature extraction also helps in reducing the size if the image which is very useful in case of big sized images.

2.4. Background Subtraction

The background subtraction method by Horprasert et al (1999), was able to cope with local illumination changes, such as shadows and highlights, even globe illumination changes. In this method, the background model was statistically modelled on each pixel. Computational colour mode, include the brightness distortion and the chromaticity distortion which was used to distinguish shading

background from the ordinary background or moving foreground objects. The background and foreground subtraction method used the following approach. A pixel was modelled by a 4-tuple [Ei, si, ai, bi], where Ei- a vector with expected colour value, si - a vector with the standard deviation of colour value, ai - the variation of the brightness distortion and bi was the variation of the chromaticity distortion of the ith pixel. In the next step, the difference between the background image and the current image was evaluated. Each pixel was finally classified into four categories: original 8 background, shaded background or shadow, highlighted background and moving foreground object. Liyuan Li et al (2003), contributed a method for detecting foreground objects in non-stationary complex environments containing moving background objects. A Bayes decision rule was used for classification of background and foreground changes based on inter-frame colour co-occurrence statistics. An approach to store and fast retrieve colour cooccurrence statistics was also established. In this method, foreground objects were detected in two steps. First, both the foreground and the background changes are extracted using background subtraction and temporal differencing. The frequent background changes were then recognized using the Bayes decision rule based on the learned colour co-occurrence statistics. Both short-term and long term strategies to learn the frequent background changes were used. An algorithm focused on obtaining the stationary foreground regions as said by Álvaro Bayona et al (2010), which was useful for applications like the detection of abandoned/stolen objects and parked vehicles. This algorithm mainly used two steps. Firstly, a sub-sampling scheme based on background subtraction techniques was implemented to obtain stationary foreground regions. This detects foreground changes at different time instants in the same pixel locations. This was done by using a Gaussian distribution function. Secondly, some modifications were introduced on this base algorithm

such as thresh holding the previously computed subtraction. The main purpose of this algorithm was reducing the amount of stationary foreground detected.

3. IMPLEMENTATION AND CONCLUSION

3.1. Implementation

3.1.1. COCO Model

3.1.1.1. Importing TensorFlow and making the hardware type to GPU.

```
[ ]  import tensorflow as tf
     device_name = tf.test.gpu_device_name()
     if device_name != '/device:GPU:0':
       raise SystemError('GPU device not found')
     print('Found GPU at: {}'.format(device_name))
```

```
The default version of TensorFlow in Colab will soon switch to
We recommend you upgrade now or ensure your notebook w
Found GPU at: /device:GPU:0
```

Fig. 6 TensorFlow and GPU

3.1.1.2. Installing basic required pycocotools

```
[ ]  !pip install Cython
     Requirement already satisfied: Cython in /usr/loc
[ ]  !ls
     sample_data
[ ]  !git clone https://github.com/waleedka/coco
     Cloning into 'coco'...
     remote: Enumerating objects: 904, done.
     remote: Total 904 (delta 0), reused 0 (delta 0),
     Receiving objects: 100% (904/904), 10.39 MiB | 26
     Resolving deltas: 100% (539/539), done.
[ ]  !ls
     coco  sample_data
[ ]  !pip install -U setuptools
     !pip install -U wheel
     !make install -C coco/PythonAPI
```

Fig. 7 Pycocotools

3.1.1.3. Installing Keras

```
[ ]  !pip install 'keras==2.1.6' --force-reinstall
```

Fig. 8 Keras

3.1.1.4. Cloning to Mask-RCNN.

```
[ ]  !git clone https://github.com/matterport/Mask_RCNN

     Cloning into 'Mask_RCNN'...
     remote: Enumerating objects: 956, done.
     remote: Total 956 (delta 0), reused 0 (delta 0), pack-reused 956
     Receiving objects: 100% (956/956), 111.84 MiB | 42.48 MiB/s, done.
     Resolving deltas: 100% (569/569), done.
```

Fig. 9 Mask-RCNN

3.1.1.5. Downloading the weights.

```
[ ]  import os
     os.chdir('./Mask_RCNN')
     !git checkout 555126ee899a144ceff09e90b5b2cf46c321200c
     !wget https://github.com/matterport/Mask_RCNN/releases/download/v2.0/mask_rcnn_coco.h5
```

Fig. 10 COCO Weights

3.1.1.6. Code to detect and segment object.

3.1.1.7. Creating model and loading weights.

```
[ ]  # Create model object in inference mode.
     model = modellib.MaskRCNN(mode="inference", model_dir=MODEL_DIR, config=config)

     # Load weights trained on MS-COCO
     model.load_weights(COCO_MODEL_PATH, by_name=True)
```

Fig. 11 Loading weights and Creating Model

3.1.1.8. Providing class name with indexes.

3.1.1.9. Running the code for Object Detection.

```
[ ]  # Load a random image from the images folder

     import os
     file_names = next(os.walk(IMAGE_DIR))[2]

     image = skimage.io.imread(os.path.join(IMAGE_DIR, random.choice(file_names)))

     # Run detection
     results = model.detect([image], verbose=1)

     # Visualize results
     r = results[0]
     visualize.display_instances(image, r['rois'], r['masks'], r['class_ids'],
                                 class_names, r['scores'])
```

Fig. 12 Object Detection

3.1.1.10. Output:-

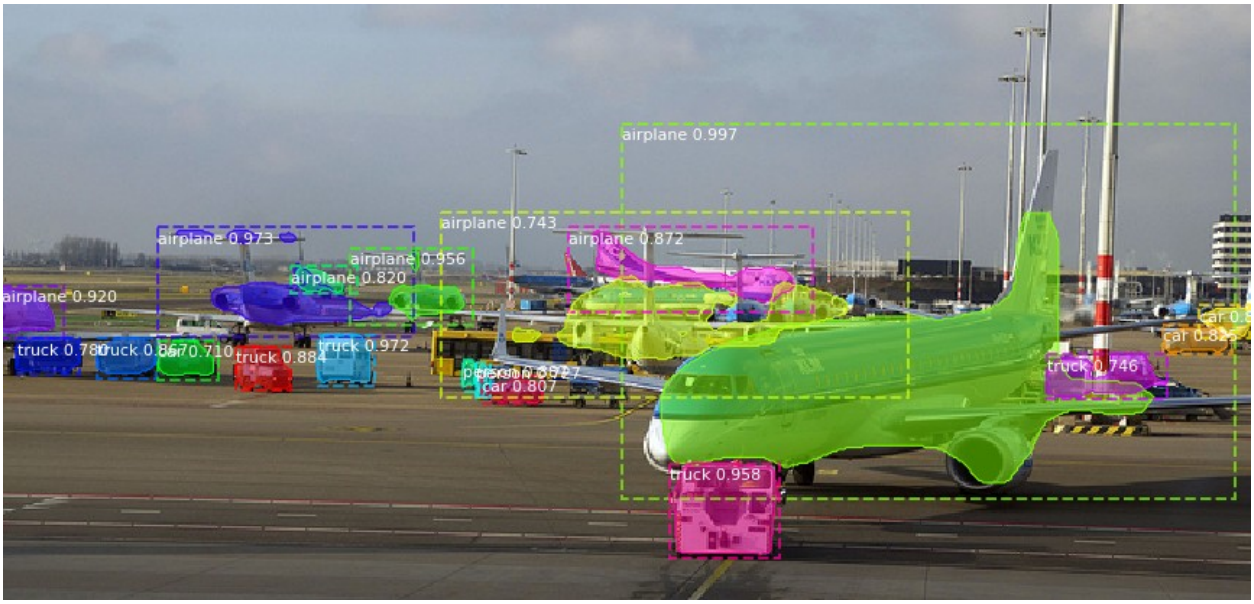

Fig. 13 Output

3.1.2. YOLO Model

3.1.2.1. Cloning the Darknet.

```
!git clone https://github.com/AlexeyAB/darknet
```

Fig. 14 Darknet

3.1.2.2. Installing CUDA and OpenCV.

```
!/usr/local/cuda/bin/nvcc --version
```

```
!apt-get install libopencv-dev


%cd darknet

!ls
!sed -i 's/OPENCV=0/OPENCV=1/g' Makefile
!sed -i 's/GPU=0/GPU=1/g' Makefile


!ls
%cd ../
!ls
```

Fig. 15 CUDA and OpenCV

3.1.2.3. Importing TensorFlow.

```
import tensorflow as tf
device_name = tf.test.gpu_device_name()
print(device_name)
```

Fig. 16 TensorFlow

3.1.2.4. Loading YOLO Weights.

```
!wget https://pjreddie.com/media/files/yolov3.weights
```

Fig. 17 YOLO Weights

3.1.2.5. Specifying a particular image.

```
!./darknet detect cfg/yolov3.cfg yolov3.weights data/person.jpg
```

Fig. 18 Image Specification

3.1.2.6. Performing Code.

3.1.2.7. Output.

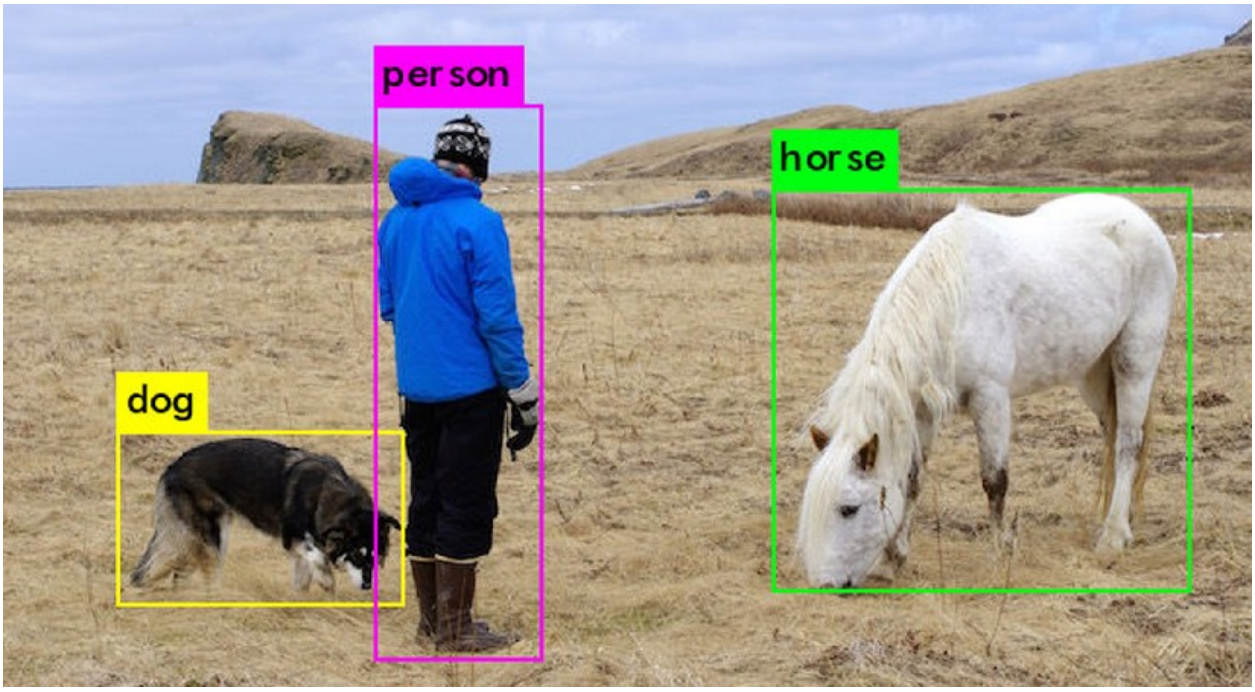

Fig. 19 Output

3.1.3. Processing a Video

3.1.3.1. Download the video

```
[ ]  !mkdir videos
     #!wget https://motchallenge.net/movies/MOT17-13.mp4 -P ./videos
     !wget https://github.com/Tony607/blog_statics/releases/download/v1.0/trailer1.mp4 -P ./videos
```

Fig. 20 Download video

3.1.3.2. Check whether video is downloaded or not

```
[ ]  !ls ./videos
```

Fig. 21 Configure

3.1.3.3. Import required libraries

```
import cv2
import numpy as np
```

Fig. 22 Libraries

3.1.3.4. Apply mask , Load Weights and specify the Classes.

```
model.load_weights(COCO_MODEL_PATH, by_name=True)


class_names = [
        'BG', 'person', 'bicycle', 'car', 'motorcycle', 'airplane',
        'bus', 'train', 'truck', 'boat', 'traffic light',
        'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird',
        'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear',
        'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie',
        'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball',
        'kite', 'baseball bat', 'baseball glove', 'skateboard',
        'surfboard', 'tennis racket', 'bottle', 'wine glass', 'cup',
        'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple',
        'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza',
        'donut', 'cake', 'chair', 'couch', 'potted plant', 'bed',
        'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote',
        'keyboard', 'cell phone', 'microwave', 'oven', 'toaster',
        'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors',
        'teddy bear', 'hair drier', 'toothbrush'
    ]
```

Fig. 23 Specifying Class

3.1.3.5. Save each frame of a video.

```
0.jpg    179.jpg  257.jpg  335.jpg  413.jpg  492.jpg  570.jpg  649.jpg  727.jpg
100.jpg  17.jpg   258.jpg  336.jpg  414.jpg  493.jpg  571.jpg  64.jpg   728.jpg
101.jpg  180.jpg  259.jpg  337.jpg  415.jpg  494.jpg  572.jpg  650.jpg  729.jpg
102.jpg  181.jpg  25.jpg   338.jpg  416.jpg  495.jpg  573.jpg  651.jpg  72.jpg
103.jpg  182.jpg  260.jpg  339.jpg  417.jpg  496.jpg  574.jpg  652.jpg  730.jpg
104.jpg  183.jpg  261.jpg  33.jpg   418.jpg  497.jpg  575.jpg  653.jpg  731.jpg
105.jpg  184.jpg  262.jpg  340.jpg  419.jpg  498.jpg  576.jpg  654.jpg  732.jpg
106.jpg  185.jpg  263.jpg  341.jpg  41.jpg   499.jpg  577.jpg  655.jpg  733.jpg
107.jpg  186.jpg  264.jpg  342.jpg  420.jpg  49.jpg   578.jpg  656.jpg  734.jpg
108.jpg  187.jpg  265.jpg  343.jpg  421.jpg  4.jpg    579.jpg  657.jpg  735.jpg
109.jpg  188.jpg  266.jpg  344.jpg  422.jpg  500.jpg  57.jpg   658.jpg  736.jpg
10.jpg   189.jpg  267.jpg  345.jpg  423.jpg  501.jpg  580.jpg  659.jpg  737.jpg
110.jpg  18.jpg   268.jpg  346.jpg  424.jpg  502.jpg  581.jpg  65.jpg   738.jpg
111.jpg  190.jpg  269.jpg  347.jpg  425.jpg  503.jpg  582.jpg  660.jpg  739.jpg
112.jpg  191.jpg  26.jpg   348.jpg  426.jpg  504.jpg  583.jpg  661.jpg  73.jpg
113.jpg  192.jpg  270.jpg  349.jpg  427.jpg  505.jpg  584.jpg  662.jpg  740.jpg
114.jpg  193.jpg  271.jpg  34.jpg   428.jpg  506.jpg  585.jpg  663.jpg  741.jpg
115.jpg  194.jpg  272.jpg  350.jpg  429.jpg  507.jpg  586.jpg  664.jpg  742.jpg
116.jpg  195.jpg  273.jpg  351.jpg  42.jpg   508.jpg  587.jpg  665.jpg  743.jpg
117.jpg  196.jpg  274.jpg  352.jpg  430.jpg  509.jpg  588.jpg  666.jpg  744.jpg
118.jpg  197.jpg  275.jpg  353.jpg  431.jpg  50.jpg   589.jpg  667.jpg  745.jpg
119.jpg  198.jpg  276.jpg  354.jpg  432.jpg  510.jpg  58.jpg   668.jpg  746.jpg
11.jpg   199.jpg  277.jpg  355.jpg  433.jpg  511.jpg  590.jpg  669.jpg  747.jpg
120.jpg  19.jpg   278.jpg  356.jpg  434.jpg  512.jpg  591.jpg  66.jpg   748.jpg
121.jpg  1.jpg    279.jpg  357.jpg  435.jpg  513.jpg  592.jpg  670.jpg  749.jpg
```

Fig. 24 Saving each frame of video

3.1.3.6. Download output video.

```
[ ]  from google.colab import files
     files.download('videos/out2.mp4')
```

Fig. 25 Output

3.2. Conclusion

COCO Model and YOLO Model are very easy to understand and work on. Once you understand the concept of object detection then the whole process becomes very simple and then you only have to focus on the steps to be taken for making it possible in real life. Here pertained dataset plays a very crucial role in Object Detection as they make our work easy , because creating your own data set is a very tedious task or you can say an another project because we have to apply various features on that like first of all collecting the images then distinguishing them in testing and training phase , then classifying each image and much more. One thing is also noticed is that we can create this project on hardware also and if possible we will try to create our own dataset and then perform this operation on real time images. And also when compared with other detection system we observed that The system which we have created is far more efficient and better than RCNN because RCNN detect object only on the basis of bounding boxes means it create a large number of bounding boxes there by making the task more tedious as it uses selective search and detect the object with different window sizes. For example if RCNN is making 1000 bounding boxes for a task to complete then our system will create 100 to do the same task.

4. REFERENCES

[1] Sandeep Kumar, Aman Balyan, Manvi Chawla. “Object Detection and Recognition in Images”. In IJEDR,2017.

[2] Khushboo Khurana, Reetu Awasthi. “Techniques for Object Recognition in Images and Multi-Object Detection”. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Volume 2, Issue 4, April 2013.

[3] Mukesh Tiwari, “A Review of Detection and Tracking of Object from Image and Video Sequences”. International Journal of Computational Intelligence Research, Volume 13, Number 5 (2017).

[4] Kanimozhi S, Gayathri G, Mala T, “Multiple Real-time object identification using Single shot Multi-Box detection”. Second International Conference on Computational Intelligence in Data Science(ICCIDS-2019).

[5] R. Girshick. “Fast r-cnn”, In Proceedings of the IEEE International Conference on Computer Vision, pages 1440–1448, 2015.

[6] Hideaki Yanagisawa, Takuro Yamashita, Hiroshi Watanabe, “A Study on Object Detection Method from Manga Images using CNN”, In IEEE, 2018.

[7] C. H. Kung, C. M. Kung, J. P. Wang, “Instrumentation and Measurement Technology Conference - IMTC 2007Warsaw, Poland, May 1-3, 2007.

[8] G. Li, R. Zeng, and L. Lin, "Moving Target Detection in Video Monitoring System", Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on, vol. 2, pp. 9778 - 9781,21-23 June 2006.

[9] Juan M. Shchez, Xavier Binefa, John R. Kender, “MULTIPLE FEATURE TEMPORAL MODELS FOR OBJECT DETECTION IN VIDEO”, In IEEE,2002.

[10] Souhail Guennouni, Anass Mansouri, Ali Ahaitouf, “Multiple Object Detection using OpenCV on an Embedded Platform”, In IEEE, 2014.