REAL-TIME MULTIPLE OBJECT DETECTION

Mini project is submitted in partial fulfillment of the requirements for the award of the degree of ***Master of Computer Applications*** *of the* ***APJ Abdul Kalam Technological University***

submitted by

ANJU BABU (TCR20MCA-2009)

DEPARTMENT OF COMPUTER APPLICATIONS
GOVERNMENT ENGINEERING COLLEGE
THRISSUR - 680009

February 2022

DEPARTMENT OF COMPUTER APPLICATIONS
GOVERNMENT ENGINEERING COLLEGE, THRISSUR
THRISSUR, KERALA STATE, PIN 680009

CERTIFICATE

This is to certify that the mini project titled **"REAL-TIME MULTIPLE OBJECT DETECTION"** *is a bonafide work done by*

ANJU BABU (TCR20MCA-2009)

under my supervision and guidance, and is submitted in February 2022 in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications from APJ Abdul Kalam Technological University(KTU).

Asst.Prof.Husain Ahamed P	Dr. Sminesh C N
Project Guide	**Head of Department**
	& Project Coordinator

Place : THRISSUR
Date :

DECLARATION

We hereby declare that the main project named, **REAL-TIME MULTIPLE OBJECT DETECTION** , is our own work and that, to the best of my knowledge and belief, it contains no material previously published another person nor material which has been accepted for the award of any other degree or course of the university or any other institute of higher learning, except where due acknowledgement and reference has been made in the text.

Signature

ANJU BABU **(TCR20MCA-2009)**

Place : THRISSUR

Date :

ACKNOWLEDGEMENT

We would like to thank Computer Application Department of GEC Thrissur, for giving us this opportunity to pursue this project and successfully complete it.

We are highly indebted to our Head of Department **Dr. Sminesh C N** for his guidance and constant supervision as well as for providing necessary information regarding the project and also for his support in completing the project.

We would like to express our deep gratitude to our project guide **Asst.Prof. Husain Ahamed P** for her timely suggestions and encouragement and guiding through the processes involved in the presentation of the project and the preparation of the Report.

We express our heart-felt gratitude to project coordinator, **Dr. Sminesh C N**, for his committed guidance, valuable suggestions, constructive criticisms and precious time that he invested throughout the work.

We sincerely thank all other faculties of MCA department for guiding through the processes involved in the project .

ABSTRACT

In recent years, deep learning has been used in image classification, object tracking, action recognition and scene labeling. Traditionally, Image Processing techniques were used to solve any Computer Vision problems occurred in an artificial intelligence system. However, in real-time identification, image processing cannot be used. This is where Deep Learning concepts are applied. We built a simple Convolutional Neural Network for object detection.The model is trained and multiple test cases are implemented in the TensorFlow environment so as to obtain accurate results. The goal of object detection is to replicate this intelligence using a computer. Our project is developed explicitly using python and Tensorflow along with other libraries. Our model is effective in detecting objects from live camera data

CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

The applications and widespread use of machine learning algorithms have made a significant change in the way we perceive computer vision problems.With the introduction of deep learning into the field of image classification, the dynamics of real-time object detection have faced a great impact.

In deep learning, the mapping is done by using representation-learning algorithms. These representations are expressed in terms of other, simpler representations. In other words, a deep learning system can represent the concept of an image for an object by combining simpler concepts, such as points and lines, which are in turn defined in terms of edges. By using a variety of algorithms, a benchmarking dataset and correct labeling packages a system can be trained to achieve the desired output. A fundamental aspect of deep learning in image classification is the use of Convolutional architectures.

To achieve the objective,the objects are detected in real-time using InceptionResNetv2 model based on Faster-RCNN

1.1 Motivation

The basic motivation behind the topic is that it is something that will over do all the physical tasks.It can be applied in different areas such as agricultural field,self driving car and area of security.The information from object detector can be used for obstacle avoidance and other interactions with the environment.

1.2 Objective

Object detection has been used in many applications,like consumer electronics(mobile),security(recognition, The above-mentioned applications have different requirements but all are coming under the object detection area. generally most of the systems can only single object class from a restricted set of views and poses need for object detection is gaining more importance. In our model, it can perform at a great accuracy using minimal hardware utilization using inceptions multi-layered CNN. tracking),photo management , assisting driving.

1.3 Contribution

The major contributions in this project are:

1. Designed and Developed an application for Real-Time object detection.

CHAPTER 2
EXISTING SYSTEM

Traditional computer vision: Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos, and other visual inputs — and take actions or make recommendations based on that information. traditional vision systems involve a human telling a machine what should be there. Computer vision is a subset of machine learning that deals with making computers or machines understand human actions, behaviors, and languages similarly to humans. Computer Vision Is Difficult Because Hardware Limits It.

Rapid progressions in Deep Learning and improvements in device capabilities including computing power, memory capacity, power consumption, image sensor resolution, and optics have improved the performance and cost-effectiveness of further quickened the spread of vision-based applications. Compared to traditional CV techniques, Deep Learning enables CV engineers to achieve greater accuracy in tasks such as image classification, semantic segmentation, object detection and Simultaneous Localization and Mapping (SLAM). Since neural networks used in Deep Learning are trained rather than programmed, applications using this approach often require less expert analysis and fine-tuning and exploit the tremendous amount of video data available in today's systems. Deep Learning also provides superior flexibility because CNN models and frameworks can be re-trained using a custom dataset for any use case, contrary to CV algorithms, which tend to be more domain-specific.

CHAPTER 3
ENVIRONMENTAL STUDY

3.1 System Configuration

System configuration describe the hardware and software requirement of the system for development

3.1.1 Hardware Requirements

- Memory : 4 GB of RAM
- Processor : Intel Core i5 or equivalent CPU
- GPU support: CUDA enabled graphics card
- Speed : 2.4 GHz
- Proper Internet Connection

3.1.2 Software Requirements

- Operating system : Windows
- Front End : Python,HTML
- IDE Used : VS Code,Google Colab
- Libraries : Numpy, Tensorflow, Tensorflow-hub,OpenCV

3.2 Software Specification

3.2.1 Python

Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant white space. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

3.2.2 HTML

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

3.2.3 VS Code

Visual Studio Code (famously known as VS Code) is a free open source text editor by Microsoft. VS Code is available for Windows, Linux, and macOS. Although the editor is relatively lightweight, it includes some powerful features that have made VS Code one of the most popular development environment tools in recent times.VS Code supports a wide array of programming languages from Java, C++, and Python to CSS, Go, and Dockerfile. Moreover, VS Code allows you to add on and even creating new extensions including code linters, debuggers, and cloud and web development support.

3.2.4 Google Colab

Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook

3.2.5 TensorFlow

TensorFlow is an open-source framework developed by Google researchers to use machine learning, deep learning, and other analytical and statistical work elements. Like similar platforms, it was designed to simplify the process of designing and implementing advanced analytic programs for users such as data scientists. And typical programmers and forecasters. TensorFlow handles data sets that are placed as graphical interfaces, and the edges connecting the nodes to the graph can represent vectors or multidimensional matrices, forming what is known as tensors.

3.2.6 Tensorflow-hub

TensorFlow Hub is an open repository and library for reusable machine learning. The tfhub.dev repository provides many pre-trained models: text embeddings, image classification models, TF.js/TFLite models and much more. The repository is open to community contributors.

3.2.7 OpenCV

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features vector space and perform mathematical operations on these features.

3.2.8 Numpy

NumPy is a library for Python. It adds support for large matrices and multidimensional arrays, along with a large collection of high-level mathematical functions to operate on these arrays. NumPy in Python gives functionality comparable to MATLAB since they are both interpreted. They allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.

CHAPTER 4
SYSTEM DESIGN

4.1 Workflow

In this project, we are using InceptionResNetV2 model based on Faster-RCNN to detect objects higher accuracy.

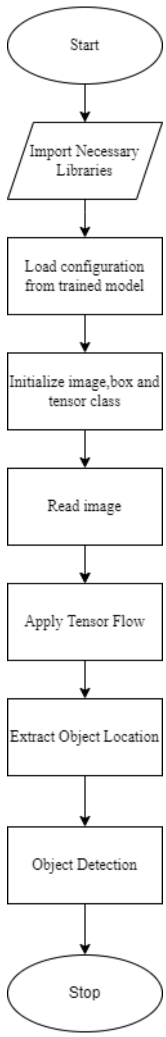

Fig. 4.1: Workflow Diagram

The implementation of our project will have 3 steps:

1. Detect object in the frame.
2. Classification of the objects.
3. Detecting objects with its accuracy.

The Workflow diagram of the system is given in Figure 4.1

4.2 Methodology

4.2.1 Introduction

This section presents our proposed approach for detecting the objects in real-time from images by using convolutional neural network deep learning process. The previous algorithms such as CNN, faster CNN, and YOLO, are only suitable for highly powerful computing machines and they require a large amount of time to train. In this project, we have tried to overcome the limitations of other algorithms by using a pre-trained model based on Faster RCNN algorithm.The proposed scheme uses Faster RCNN algorithm for detection with higher accuracy. Specifically, our proposed approach uses a new architecture as a combination of multilayer of convolutional neural network. The algorithm comprises of two phases. First, it reduces the feature maps extraction of spatial dimensions by using resolution multiplier. Second, it is designed with the application of small convolutional filters for detecting objects by using the best aspect ratio values. The major objective during the training is to get a high-class confidence score by matching the default boxes with the ground truth boxes. The advantage of having multi-box on multiple layers leads to significant results in detection.

4.2.2 Data Set

OPenImages v4 Open Images V4, a dataset of 9.2M images with unified annotations for image classification, object detection and visual relationship detection. Open Images V4 offers large scale across several dimensions:

30.1M image-level labels for 19.8k concepts, 15.4M bounding boxes for 600 object classes, and 375k visual relationship annotations involving 57 classes. For object detection in particular, it provide 15x more bounding boxes than the next largest datasets (15.4M boxes on 1.9M images). The images often show complex scenes with several objects (8 annotated objects per image on average). **coco2017** COCO is a large-scale object detection, segmentation, and captioning dataset of many object types easily recognizable by a 4-year-old. The data is initially collected and published by Microsoft.

4.2.3 CNN

A Convolutional neural network is a neural network that has one or more convolutional layers and are used mainly for image processing,classification,segmentation. The first layer is the input layer wherein the entire dataset(in batches) is fed into the network model. The process of feature extraction begins from the second layer until it reaches the last image in the last batch. A different feature is extracted in each layer as the construction of the neural network progresses. For example, first it extracts a point, then a line and then a curve. All the features are combined at the end (fully-connected layer) resulting in an output layer.

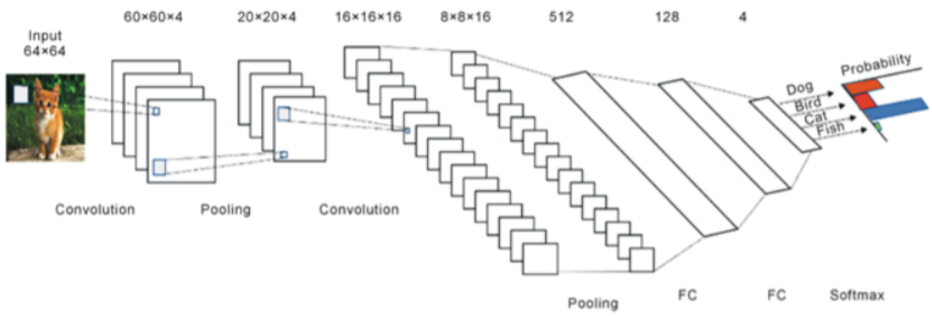

Fig. 4.2: CNN Architecture.

4.2.4 Faster RCNN

Faster R-CNN is an object detection model that improves on Fast R-CNN by utilising a region proposal network (RPN) with the CNN model. The RPN shares full-image convolutional features with the detection network, enabling nearly cost-free region proposals. It is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. The RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection. RPN and Fast R-CNN are merged into a single network by sharing their convolutional features: the RPN component tells the unified network where to look.

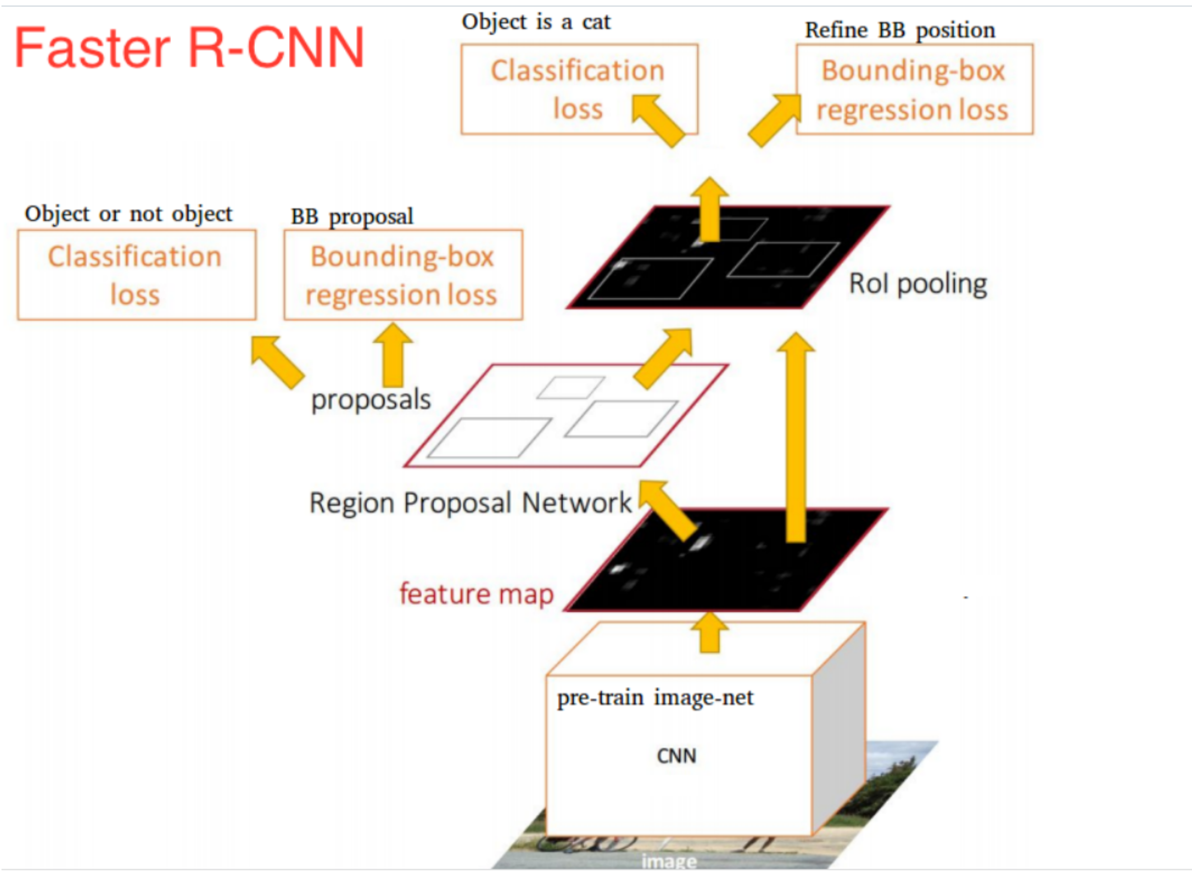

Fig. 4.3: Faster RCNN Architecture.

4.2.5 Flow Chart

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.

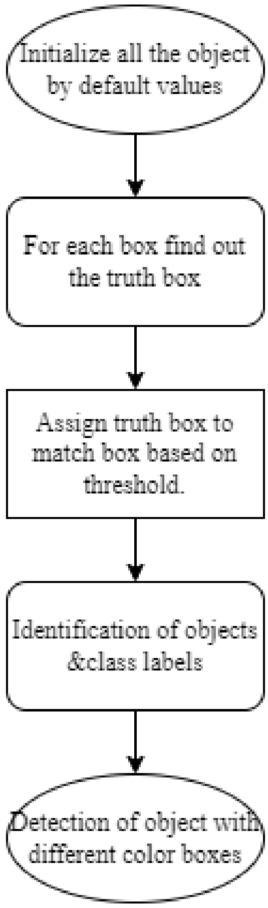

Fig. 4.4: Flow chart

4.3 Implementation

In this project one of the main step is download and load the pretrained model and data set.We import useful libraries such as tensorflow,tensorflowhub.We capture live images /video and give it as input.Once the input is loaded,the model will detect available objects in the frame.Then we create the bounding boxes for each classes and print the class names on bounding boxes.Then we display the output frame to the user.We can implement this system using google Colab.

CHAPTER 5

RESULTS AND SCREENSHOTS

5.1 Source code screenshots

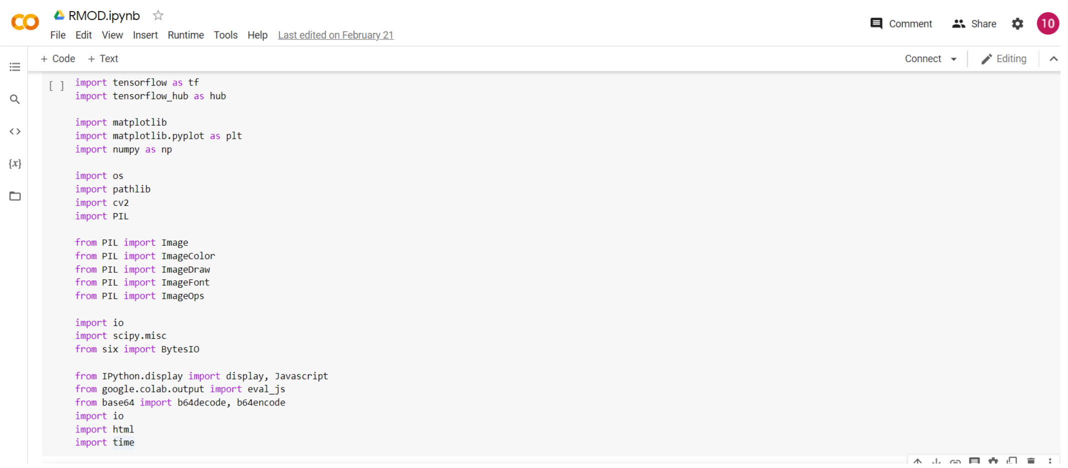

Fig. 5.1: Source code sample screenshots.1

Fig. 5.2: Source code sample screenshots.2

Fig. 5.3: Source code sample screenshots.3

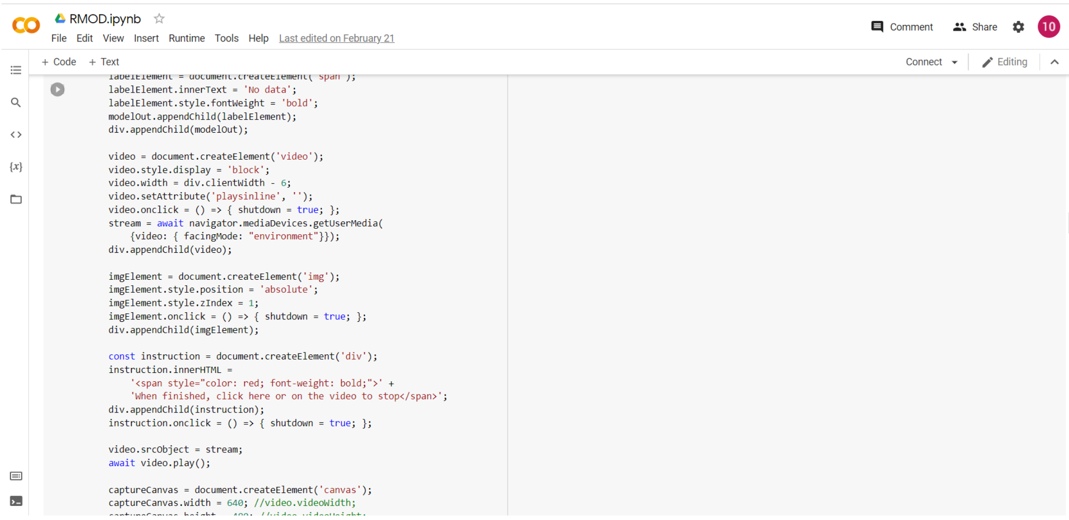

Fig. 5.4: Source code sample screenshots.4

Fig. 5.5: Source code sample screenshots.5

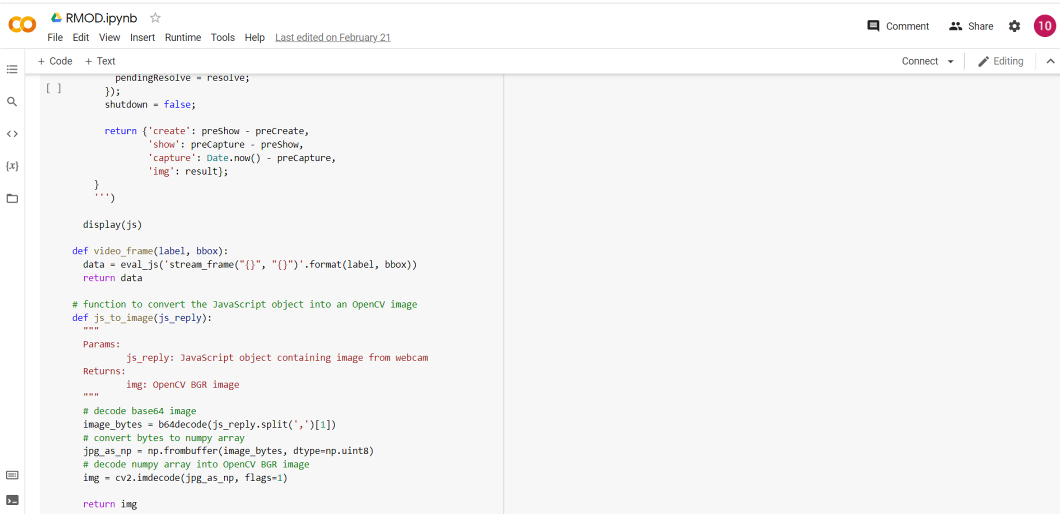

Fig. 5.6: Source code sample screenshots.6

```python
def bbox_to_bytes(bbox_array):
  """
  Params:
          bbox_array: Numpy array (pixels) containing rectangle to overlay on video stream.
  Returns:
        bytes: Base64 image byte string
  """
  # convert array into PIL image
  bbox_PIL = PIL.Image.fromarray(bbox_array) # numpy array into PIL image object
  iobuf = io.BytesIO()

  # format bbox into png for return
  bbox_PIL.save(iobuf, format='png')

  # format return string
  bbox_bytes = 'data:image/png;base64,{}'.format((str(b64encode(iobuf.getvalue()), 'utf-8')))

  return bbox_bytes
```

```python
def run_inference_for_single_image(model, image, live_cam):

    # convert image into numpy
    image = np.asarray(image)
    #print('Converted image into numpy type:', type(image))

    # The input needs to be a tensor, convert it using `tf.convert_to_tensor`.
    input_tensor = tf.convert_to_tensor(image)
    #print('Converted numpy into tensor format:', input_tensor)

    # The model expects a batch of images, so add an axis with `tf.newaxis`.
    input_tensor = input_tensor[tf.newaxis,...]
```

Fig. 5.7: Source code sample screenshots.7

```python
    # Run inference
    if not live_cam:
      start_time = time.time()
      output_dict = model(input_tensor)
      end_time = time.time()
      print(f"Inference time: {np.ceil(end_time-start_time)} seconds per frame")

    output_dict = model(input_tensor)
    num_detections = int(output_dict.pop('num_detections')) # 300

    # All outputs are batches tensors.
    # Convert to numpy arrays, and take index [0] to remove the batch dimension.
    # We're only interested in the first num_detections.

    output_dict = {key: value[0, :num_detections].numpy()
                   for key, value in output_dict.items()}

    output_dict['num_detections'] = num_detections

    # detection_classes should be ints.
    output_dict['detection_classes'] = output_dict['detection_classes'].astype(np.int64)

    return output_dict
```

Fig. 5.8: Source code sample screenshots.8

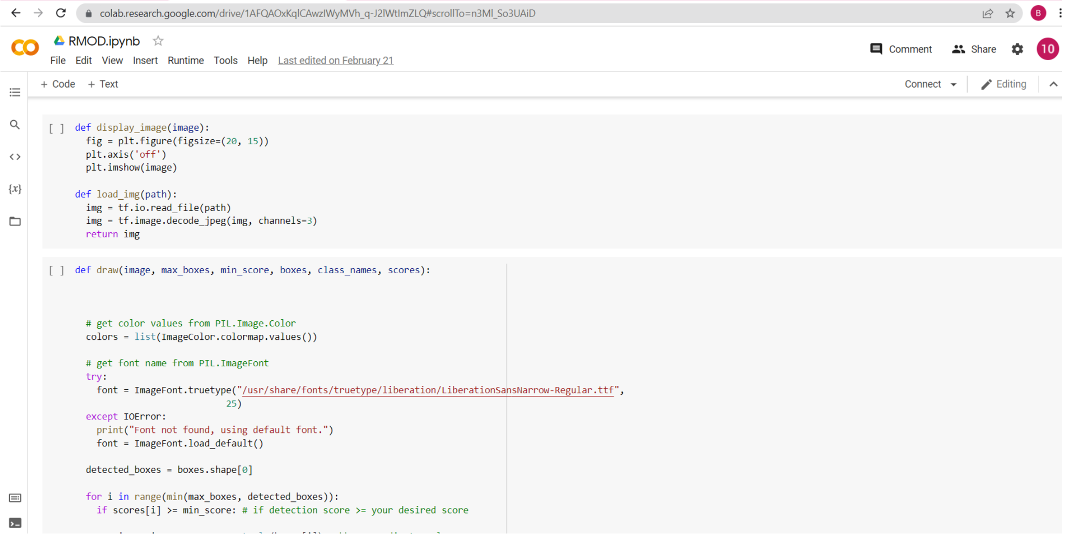

Fig. 5.9: Source code sample screenshots.9

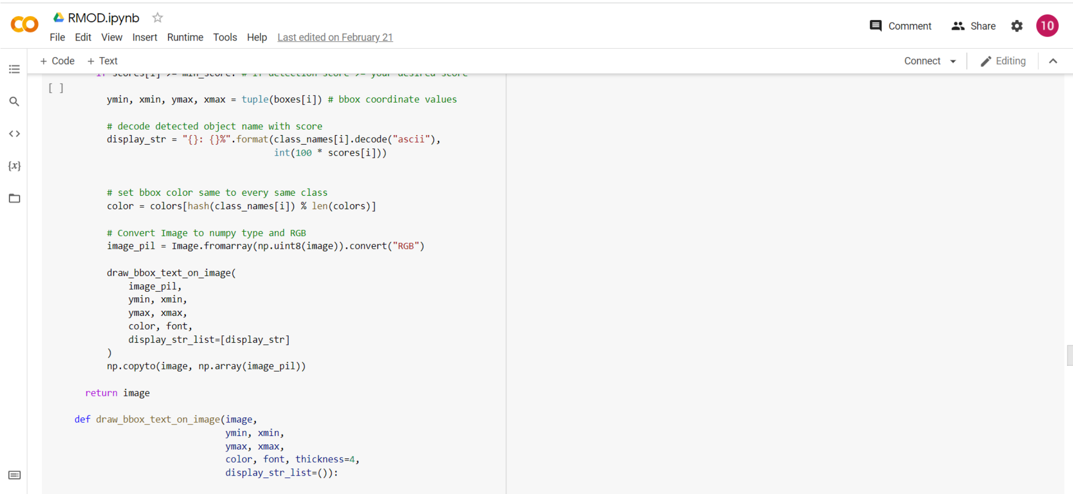

Fig. 5.10: Source code sample screenshots.10

```
# creates PIL Draw Object
draw = ImageDraw.Draw(image)
im_width, im_height = image.size

# Formula for bbox coordinate calculation
(left, right, top, bottom) = (xmin * im_width, xmax * im_width,
                              ymin * im_height, ymax * im_height)

# Draw bbox with coordinates
draw.line([(left, top), (left, bottom), (right, bottom), (right, top),
          (left, top)],
          width=thickness,
          fill=color)

# get font size from String list
display_str_heights = [font.getsize(ds)[1] for ds in display_str_list] # 11

# Each display_str has a top and bottom margin of 0.05x.
total_display_str_height = (1 + 2 * 0.05) * sum(display_str_heights) # 12


# calculation for showing better class label position with bbox
if top > total_display_str_height:
  text_bottom = top
else:
  text_bottom = top + total_display_str_height

for display_str in display_str_list:
  text_width, text_height = font.getsize(display_str)
  margin = np.ceil(0.05 * text_height)

  # draw rectangle with color
  draw.rectangle([(left, text_bottom - text_height - 2 * margin),
```

Fig. 5.11: Source code sample screenshots.11

```
  # draw rectangle with color
  draw.rectangle([(left, text_bottom - text_height - 2 * margin),
                  (left + text_width, text_bottom)],
                 fill=color)

  # put text in drawn colored rectangle
  draw.text((left + margin, text_bottom - text_height - margin),
            display_str,
            fill="black",
            font=font)

def run_detector(detector, path, max_boxes, max_score):


  img = load_img(path)

  # convert jpeg into tf.float32 fromat and add batchsize 1 as dimension
  converted_img= tf.image.convert_image_dtype(img, tf.float32)[tf.newaxis, ...]
  print('Detecting Image ...')
  output = detector(converted_img) # Returns a dictionary containing bbox values, classnames, scores
  print('Detection Complete\n')

  for key, value in output.items():
    print(f'Key:{key}\nValue:{value}\n\n')

  output = {key:value.numpy() for key,value in output.items()}
  print("Found %d objects." % len(output["detection_class_entities"]))

  image_with_boxes = draw(img.numpy(),
                          max_boxes, max_score,
                          output['detection_boxes'],
```

Fig. 5.12: Source code sample screenshots.12

Fig. 5.13: Source code sample screenshots.13

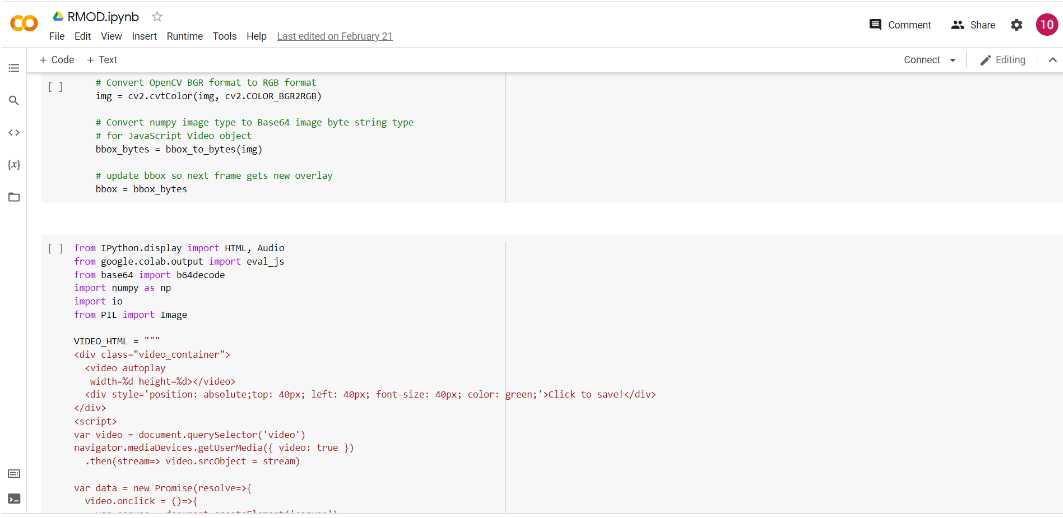

Fig. 5.14: Source code sample screenshots.14

Fig. 5.15: Source code sample screenshots.15

5.2 *Results*

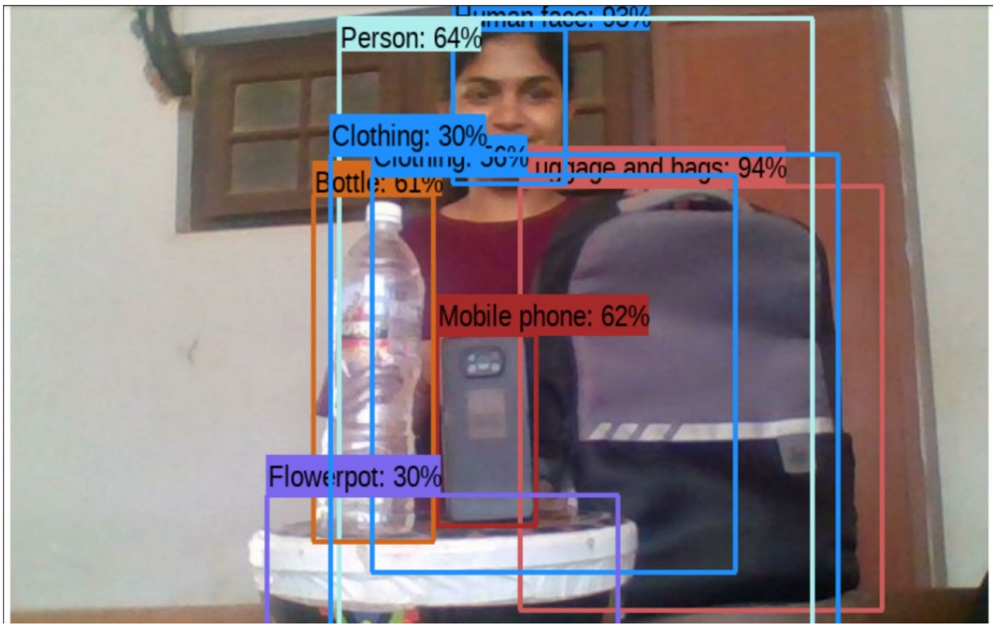

Fig. 5.16: RealTime captured image

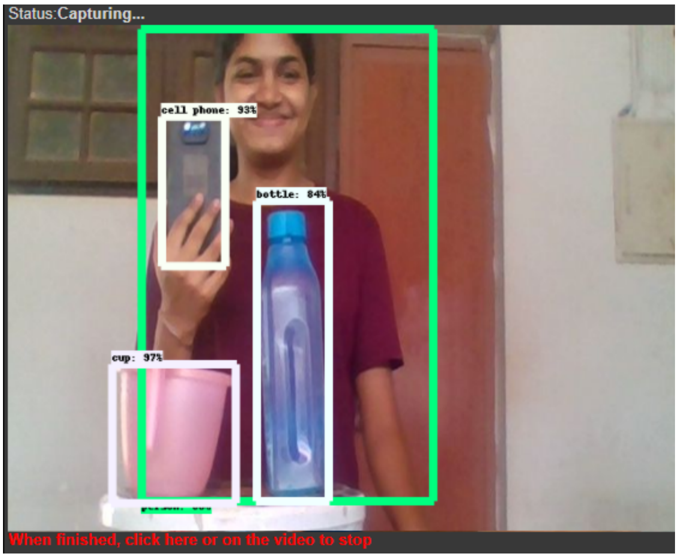

Fig. 5.17: Live cam

In the object detection from image section ,the system will capture live image and detect the objects within the image frame.In the output screen the objects will be shown inside a bounding box and the class label and accuracy will be printed on it.

In the object detection from live camera section ,the system will detect objects within the live video frame.In the output screen the objects will be shown inside a bounding box and the class label and accuracy will be printed on it.

CHAPTER 6
CONCLUSION

Object detection is a key ability for most computer and robot vision system. Although great progress has been observed in the last years, and some existing techniques are now part of many consumer electronics (e.g., face detection for auto-focus in smartphones) or have been integrated in assistant driving technologies, we are still far from achieving human-level performance, in particular in terms of open-world learning. It should be noted that object detection has not been used much in many areas where it could be of great help. As mobile robots, and in general autonomous machines, are starting to be more widely deployed (e.g., quad-copters, drones and soon service robots), the need of object detection systems is gaining more importance. Finally, we need to consider that we will need object detection systems for nano-robots or for robots that will explore areas that have not been seen by humans, such as depth parts of the sea or other planets, and the detection systems will have to learn to new object classes as they are encountered. In such cases, a real-time open-world learning ability will be critical

CHAPTER 7
REFERENCES

[1]Real-Time Object Detection using TensorFlow(IJCRT) 1Rinkesh U Patel, 2Meet S Patel, 3Dev A Thakkar, 4Bhumika Bhatt https://ijcrt.org/papers/IJCRT192261.pdf

[2] Real-Time Object Detection using TensorFlow(IRJET) Priyal Jawale, Hitiksha Patel Chaudhary2, Nivedita Rajput3 Learning Techniques" *In 2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS)* , (pp. 80-85).

[3]https://www.mygreatlearning.com/blog/object-detection-using-tensorflow/