Comparison of Object Detection Models

BTech-CP Semester- VI

Prepared at

ISO 9001:2008
ISO 27001:2013
CMMI LEVEL-5

Bhaskaracharya National Institute for Space Applications & Geo-informatics
Ministry of Electronics and Information Technology, Govt. of India.

Gandhinagar

Prepared By

Tanmay Vaidya.
ID No. H4

Deepnil Vasava.
ID No. H4

Deep Patel.
ID No. H4

Guided By:
Prof. Mayur Vegad
Department of Computer Engineering
BVM, Anand.

External Guide:
Vishal Patel
Project Scientist
BISAG- N, Gandhinagar.

SUBMITTED TO

Birla Vishvakarma Mahavidyalaya

Engineering College, Vallabh Vidyanagar

[An Autonomous Institution]

Managed by **Charutar Vidyamandal**

Bhaskaracharya National Institute for Space Applications and Geo-informatics
MeitY, Government of India
Phone: 079 - 23213081 Fax: 079 - 23213091
E-mail: info@bisag.gujarat.gov.in, website: https://bisag-n.in/

BISAG-N
ISO 9001:2008
ISO 27001:2013
CMMI LEVEL-5

CERTIFICATE

This is to certify that the project report compiled by ***Mr. Tanmay Vaidya, Mr. Deepnil Vasava and Mr. Deep Patel*** *students of 7th Semester* ***BTech -CP from Birla Vishvakarma Mahavidyalaya, Anand*** *have completed their summer internship project satisfactorily. To the best of our knowledge this is an original and bonafide work done by them. They have worked on Research-based project for "Comparison of Object detection models", starting from June 4th, 2021 to July 3rd , 2021.*

During their tenure at this Institute, they were found to be sincere and meticulous in their work. We appreciate their enthusiasm & dedication towards the work assigned to them.

We wish them every success.

Vishal Patel

Project Scientist,

BISAG- N, Gandhinagar

T. P. Singh

Director General,

BISAG- N, Gandhinagar

Birla Vishvakarma Mahavidyalaya
(Engineering College)
Post Box No. 20,
Vallabh Vidyanagar,
District: Anand. PIN 388120
Gujarat, India.

Mayur M Vegad
Professor
Department of Computer Engineering
mayurmvegad@bvmengineering.ac.in
Phone: 02692-230104

6th July, 2021

To whomsoever it may concern

We are pleased to permit **Mr. Deepnil Vasava** (18CP014), **Mr. Deep Patel** (18CP204), and **Mr. Tanmay Vaidya** (18CP013) to carry out their Internship-II (Course code: CPIS2) at Bhaskaracharya National Institute for Space Application and Geo-informatics (BISAG-N), Gandhinagar, Gujarat. This course (CPIS2) is one of the compulsory audit (non-credit) courses and is required to be completed as a partial fulfillment of the B.Tech. in Computer Engineering program at BVM Engineering College.

(Mayur M Vegad)
Faculty Coordinator for above student

About BISAG- N

ABOUT THE INSTITUTE

Modern day planning for inclusive development and growth calls for transparent, efficient, effective, responsive and low cost decision making systems involving multi-disciplinary information such that it not only encourages people's participation, ensuring equitable development but also takes into account the sustainability of natural resources. The applications of space technology and Geo–informatics have contributed significantly towards the socio-economic development. Taking cognizance of the need of geo-spatial information for developmental planning and management of resources, the department of Science and Technology, Government of Gujarat established "Bhaskaracharya National Institute for Space Applications and Geo-informatics - N" (BISAG- N). BISAG- N is an ISO 9001:2008, ISO 27001:2005 and CMMI: 5 certified institute. BISAG- N which was initially set up to carryout space technology applications, has evolved into a centre of excellence, where research and innovations are combined with the requirements of users and thus acts as a value added service provider, a technology developer and as a facilitator for providing direct benefits of space technologies to the grass root level functions/functionaries.

BISAG- N's Enduring Growth

Since its foundation, the Institute has experienced extensive growth in the sphere of Space technology and Geo-informatics. The objective with which BISAG- N was established is manifested in the extent of services it renders to almost all departments of the State. Year after year the institute has been endeavouring to increase its outreach to disseminate the use of geo-informatics up to grassroots level. In this span of nine years, BISAG- N has assumed multi-dimensional roles and achieved several milestones to become an integral part of the development process of the Gujarat State.

BISAG- N Journey

2003-04

Gujarat SATCOM Network

2007-08

Centre for Geo-informatics Applications

2010-11

Academy of Geo-informatics for Sustainable Development

2012-13

A full-fledged Campus

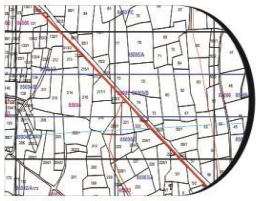
Activities

Satellite Communication..

for promotion and facilitation of the use of broadcast and teleconferencing networks for distant interactive training, education and extension.

Remote Sensing..

for Inventory, Mapping, Developmental planning and Monitoring of natural & man-made resources.

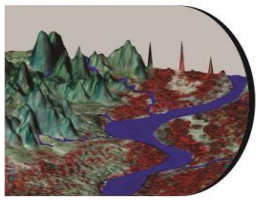


Geographic Information System..

for conceptualization, creation and organization of multi purpose common digital database for sectoral/integrated decision support systems.

Global Navigation Satellite System..

for Location based Services, Geo-referencing, Engineering Applications and Research.



Photogrammetry..

for Creation of Digital Elevation Model, Terrain Characteristic, Resource planning.

Cartography..

for thematic mapping, value added maps.

Software Development..

for wider usage of Geo-spatial applications, Decision Support Systems (desktop as well as web based), ERP solutions.

Education, Research and Training..

for providing Education, Research, Training & Technology Transfer to large number of students, end users & collaborators.

Applications of Geospatial Technology for Good Governance: Institutionalization

Through the geospatial technology, the actual situation on the ground can be accessed. The real life data collected through the technology forms the strong foundation for development of effective social welfare programs benefiting directly the grass root level people. The geospatial data collected by the space borne sensors along with powerful software support through Geographic Information System (GIS), the vital spatio-temporal maps, tables, and various statistics are being generated which feed into Decision Support System (DSS).

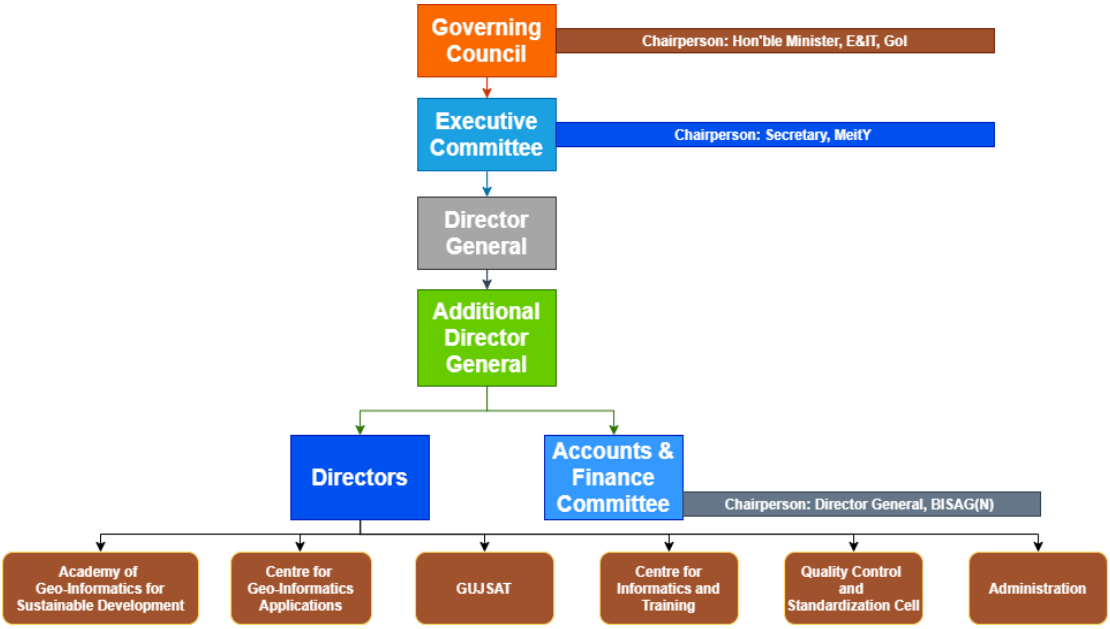
A multi-threaded approach is followed in the process of institutionalization of development of such applications. The 5 common threads which run through all the processes are: *Acceptability, Adaptability, Affordability, Availability and Assimilability*.
These are the "Watch Words" which any application developer has to meet. The "acceptability" addresses the issue that the application developed has met the wide acceptability among the users departments and the ultimate end beneficiary by way of providing all necessary data and statistics required. The "affordability" addresses the issue of the application product being cost effective. The "availability" aspect looks into aspect of easily accessible across any platform, anywhere and anytime. The applications should have inbuilt capability of easy adaptability to the changing spatio- and temporal resolutions of data, new aspects of requirements arising from time to time from users. The assimilability aspect ensures that the data from various sources / resolutions and technologies can be seamlessly integrated.

ACCEPTABILITY	▪ Problem definition by users • Proof of Concept development without financial liability on users ▪ Execution through collaboration under user's ownership
ADOPTABILITY	▪ Applications as per present systems & database ▪ Maximum Automation ▪ Minimum capacity building requirement at the user end
AFFORDABILITY :	▪ Multipurpose geo-spatial database, common, compatible, standardized (100s of layers) ▪ In house developed/open source software ▪ Full Utilization of available assets
AVAILABILITY:	▪ Departmental /Integrated DSS ▪ Desired Product delivery anytime, anywhere in the country
ASSIMILABILITY	▪ Integration of Various technologies like RS, GIS, GPS, Web MIS, Mobile etc.

Organizational Setup

The Institute is responsible for providing information and technical support to different Departments and Organizations. The Governing Body and the Empowered Executive Committee govern the functioning of BISAG- N. The Institute is registered under the Societies Registration Act 1860. Considering the scope and extent of activities of BISAG- N, its organizational structure has been charted out with defined functions.

Organizational Setup of BISAG- N



Governing Body

For smoother, easier and faster institutionalization of Remote Sensing and GIS technology, decision makers of the state were brought together to form the Governing Body. It is the supreme executive authority of the Institute. The Governing Body comprises of ex-officio members from various Government departments and Institutes.

- Hon'ble Minister of Electronics and Information Technology ..... Chairperson (Ex-Officio)
- Hon'ble Minister of State Electronics and Information Technology ..... Deputy Chairperson (Ex-Officio)
- Secretary of Government of India: Ministry of Electronics and Information Technology ..... Executive Vice Chairperson (Ex-Officio)
- Chief Executive Officer, Niti Aayog ..... Member (Ex-Officio)
- Chairman, Indian Space Research Organization ..... Member (Ex-Officio)
- Secretary to Government of India: Department of Science and Technology ..... Member (Ex-Officio)
- Additional Secretary to Government of India: Ministry of Electronics and Technology ..... Member (Ex-Officio)
- Chief Secretary to Government of Gujarat ..... Member (Ex-Officio)
- President & Chief Executive Officer, National e-Governance Division, Ministry of Electronics
- and Information Technology ..... Member (Ex-Officio)
- Financial Advisor to Government of India: Ministry of Electronics and Information Technology… Member (Ex-Officio)
- Distinguished Professionals from the GIS field-Three (3) (To be nominated by the Chairperson)
- Director-General, Bhaskaracharya National Institute for Space Application and Geo-Informatics {BISAG(N)} ..... Member Secretary (Ex-Officio)

Centre for Geo-informatics Applications

Introduction

The objective of this technology group is to provide decision support to the sectoral stake holders through scientifically organized, comprehensive, multi-purpose, compatible and large scale (village level) geo-spatial databases and supporting analytical tools. These activities of this unit are executed by a well-trained team of multi-disciplinary scientists. The government has provided a modern infrastructure along with the state-of-the-art hardware and software. To study the land transformation and development over the years, a satellite digital data library of multiple sensors of last twenty years has been established and conventional data sets of departments have been co-registered with satellite data. The geo-spatial databases have been created using conventional maps, high resolution satellite 2D and 3D imagery and official datasets (attributes).The geo-spatial databases include terrain characteristics, natural and administrative systems, agriculture, water resources, city survey maps, village maps with survey numbers, water harvesting structures, water supply, irrigation, power, communications, ports, land utilization pattern, infrastructure, urbanization, environment data, forests, sanctuaries, mining areas, industries. They also include social infrastructure like the locations of schools, health centres, institutions, aganwadies, local government infrastructure etc. The geospatial database of nagar-palikas includes properties and amenities captured on city and town planning maps with 1000 GIS layers. Similar work for villages has been initiated as a pilot project.

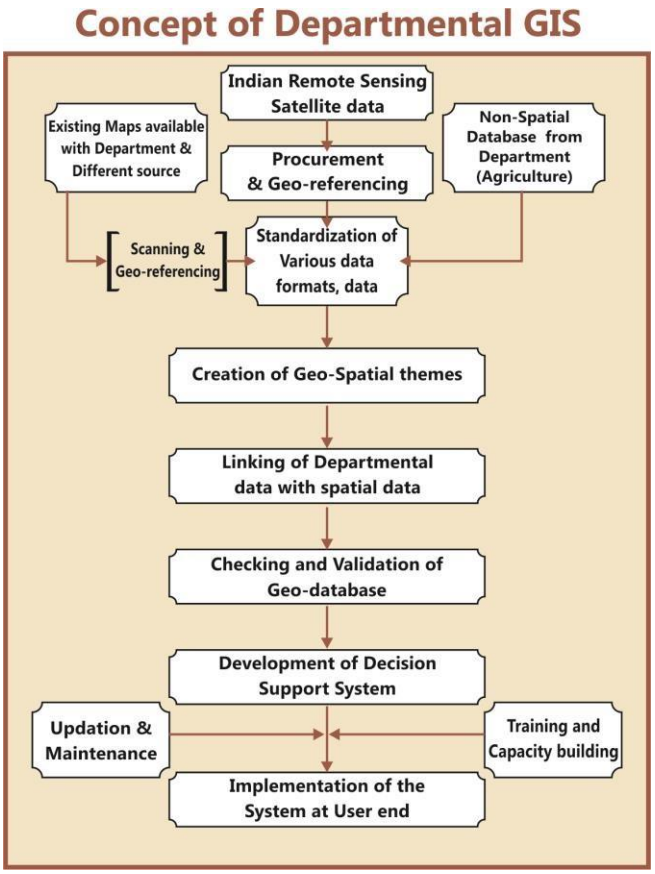
The applications of space technology and geo-informatics have been operational in almost all the development sectors of the state. Remote sensing and GIS applications have provided impetus to planning and developmental activities at grass root level as well as monitoring and management in various disciplines.

The GIS based Applications Development

The GIS software is a powerful tool to handle, manipulate and integrate both the spatial and non-spatial data. The GIS system operates on the powerful backend data base and Sequential Query Language (SQL) to inquiry the data bases. It has the capability to handle large volume of data and process to yield values of parameters which can be input to very important government activity as Decision Support System (DSS). Its mapping capabilities help the users and specialists in generating single and multi-theme wise maps.

The GIS based applications development has been institutionalized in BISAG- N. This process can be listed as (Refer Figure for Details)

- Making the users aware of the GIS capabilities through introductory training programme and by exposing to already developed projects as success stories.
- Helping the users in defining the GIS based projects.
- Digitizing the data available with the users and encouraging them to collect any additional data as may be required.
- Generating the appropriate data bases with the full involvement of the users following the data bases standards

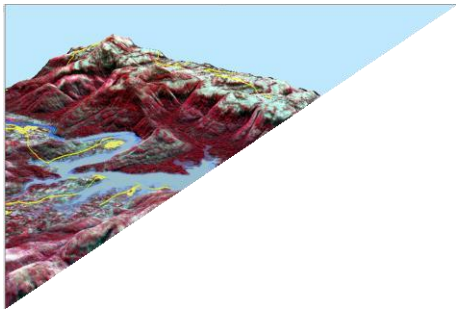


Remote Sensing and GIS Sectoral Applications:

Geo-informatics based Irrigation Management and Monitoring System

- The Geo-spatial information system for Irrigation water Management and Monitoring system for command areas in Sardar Sarovar Narmada Nigam Limited (SSNL) has been developed. Satellite image-based Irrigation monitoring system has been developed in GIS. From the multi-spectral Satellite images of every month, the irrigated areas were extracted.
- The irrigated area were overlaid on the geo-referenced cadastral maps and the statistics of area irrigated has been estimated.
- The user friendly Customized Decision Support System (DSS) has been developed.

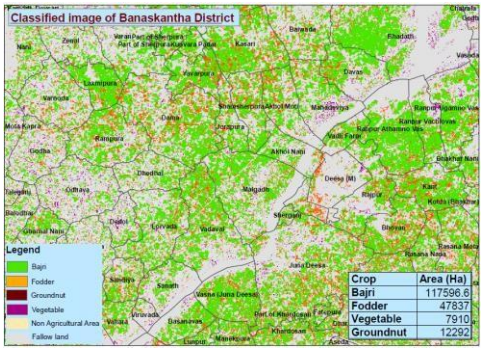
Preparation of DPR of Par–Tapi-Narmada Link using Geo-informatics for National Water development Agency (NWDA)



- The main objective of Par–Tapi-Narmada Link project is to divert surplus water available in west flowing rivers of south Gujarat and Maharashtra for utilization in the drought prone Saurashtra and Kachcha. On the request from NDWA, preparation of various maps for proposed DPR work was undertaken by the BISAG- N. Land use and submergence maps of proposed dams along with its statistics have been prepared by the BISAG- N. The detailed work consisted of generation of Digital Elevation Model (DEM), contour generation, Land use mapping, forest area generation of submergence extent at different levels etc.

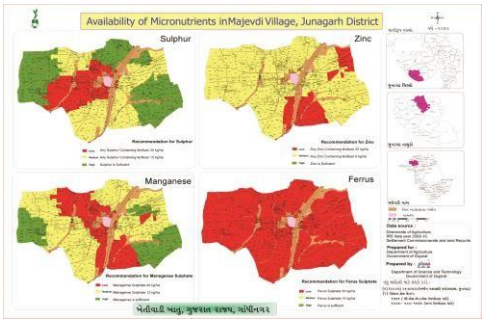
Agriculture

District and Village-level Crop Inventory



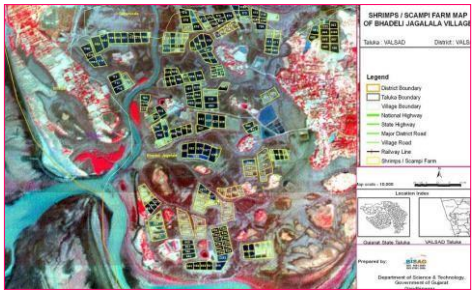
- Remote Sensing (RS) based Village-level Crop Acreage Estimation at was taken up in two villages of Anand and Mehsana districts of Gujarat state. The major objective of this study was to attempt village-level crop inventory during two crop seasons of Kharif (monsoon season) and Rabi (winter season) using single-date Indian Remote Sensing (IRS) LISS-III and LISS-IV digital data of maximum vegetative growth stage of major crops during each season.
- District-level crop acreage estimation during three cropping seasons namely Kharif, Rabi and Zaid (summer) seasons was also carried out in all the 26-districts of Gujarat State. Summer crop acreage estimation Gujarat State was carried out during 2012.

Spatial Variability Mapping of Soil Micro-Nutrients



- The spatial variability of soil micro-nutrients like Fe, Mn, Zn and Cu in various villages of different districts, Gujarat state was mapped using geo-informatics technology. The major objectives of this study were i) to quantify the variability of Mn, Fe, Cu and Zn concentration in soil; ii) to map the pattern of micro-nutrient variability in cadastral maps, iii) suggest proper application of micro-nutrients based on status of deficiency for proper crop management and iv) preparation of village-level atlases showing spatial variability of micro-nutrients.

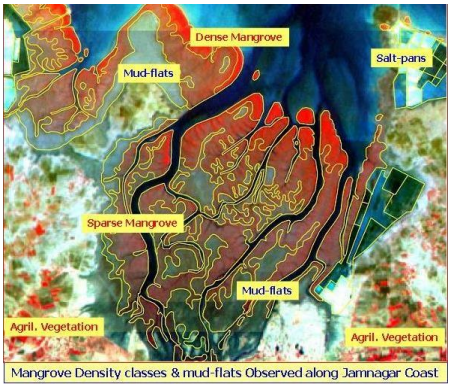
Geo-spatial Information System for Coastal Districts of Gujarat



- The project on development of Village-level Geo-spatial Information System for Shrimp Farms in Coastal Districts of Gujarat, was taken with major objective of development of Village-level Geo-spatial Information System for Shrimp/Scampi areas using Remote Sensing (RS) and GIS. This project was sponsored by the Marine Products Export Development Authority (MPEDA), Ministry of Commerce & Industry, Government of India for scientific management of Scampi farms in the coastal districts which can help fishermen to better their livelihood and increase the economic condition on sustainable basis. The customized query shell was developed using the open source software for sharing the information amongst the officers from MPEDA and potential users. This has helped the farmers to plan their processing and marketing operations so as to achieve better remunerations.

Environment and Forest

Mapping and Monitoring of Mangroves in the Coastal Districts of Gujarat State



Mangrove Density classes & mud-flats Observed along Jamnagar Coast

- Gujarat Ecology Commission, with technical inputs from the Bhaskaracharya National Institute for Space Applications and Geo-informatics - N (BISAG- N) made an attempt to publish Mangrove Atlas of the Gujarat state. Mangrove atlas for 13-coastal districts with 35-coastal talukas in Gujarat, have been prepared using Indian Remote sensing satellite images. The comparison of mangrove area estimates carried out by BISAG- N and Forest Survey of India (FSI) indicates a net increase in the area under mangrove cover. The present assessment by BISAG- N, has recorded 996.3 sq. km under mangrove cover, showing a steep rise to the tune of 88.03 sq. km. In addition to the existing Mangrove cover, the present assessment also gives the availability of potential area of 1153 sq. km, where mangrove regeneration program can be taken up.

Academy of Geo-informatics for Sustainable Development

Introduction

- Considering the requirement of high end research and development in the areas having relevance of geo-informatics technology for sustainable development, a separate infrastructure has been established. In collaboration with different institutes in the state as well as in the country, R&D activities are being carried out in the areas of climate change, environment, disaster management, natural resources management, infrastructure development, resources planning, coastal hazard and coastal zone management studies, etc. under the guidance of eminent scientists.

- Various innovative methodologies/models developed in this academy through the research process have helped in development of various applications. There are plans to enhance R&D activities manifold during coming years.
- This unit also provides training to more than 600 students every year in the field of Geo-informatics to the students from various backgrounds like water resources, urban planning, computer Engineering, IT, Agriculture in the areas of Remote sensing, GIS and their applications.

- This Academy has been established as a separate infrastructure for advanced research and development through following schools:
  - School of Geo-informatics
  - School of Climate & Environment
  - School of Integrated Coastal Zone Management

- School of Sustainable Development Studies
- School of Natural Resources and Bio-diversity
- School of Information Management of Disasters
- School of Communication and Society

During XIIth Five year Plan advance applied research through above schools shall be the main thrust area. Already M. Tech and Ph.D. students of other Universities/ Institutes are doing research in this academy in applied sciences under various collaborative programmes.

M. Tech. Students' Research Programme

The academy started M. Tech. students' research programme in a systematic way. It admitted 11 students from various colleges and universities in Gujarat, Rajasthan and Madhya Pradesh for period of 10 months from August 2011 to May 2012. All the students were paid stipend of Rs. 6000 per month during the tenure. The research covered the following areas:

- Cloud computing techniques
- Mobile communication
- Design of embedded systems
- Aquifer modelling
- Agricultural and Soils Remote Sensing
- Digital Image processing Techniques (Data Fusion and Image Classification).

The research resulted in various dissertations and publications in national and international journals.

• Now nine students, one from IIT, Kharagpur, three from GTU, one from M. S University, Vadodara and four from GU, are undergoing their Ph. D programme. Out of nine, two thesis have been submitted. Two students are from abroad. One each from Vietnam and Yemen. Since then (after approval of research programme from the Governing Body), 200+ papers have been published by the Academy.

CANDIDATE'S DECLARATION

We declare that 7th semester internship project report entitled "**Comparison of Object detection Models**" is our own work conducted under the supervision of the external guide **Vishal Patel** from BISAG-N (Bhaskaracharya National Institute for Space Applications & Geo-informatics). We further declare that to the best of our knowledge the report for this project does not contain any part of the work which has been submitted previously for such project either in this or any other institutions without proper citation.

Candidate 1's Signature

Tanmay A Vaidya

Student ID: H4

Candidate 3's Signature

Deep J Patel

Student ID: H4

Candidate 2's Signature

Deepnil K Vasava

Student ID: H4

Submitted To:

Birla Vishvakarma Mahavidyalaya Engineering College, Vallabh Vidyanagar

Managed by Charutar Vidyamandal

ACKNOWLEDGMENT

We are grateful to **T.P. Singh,** Director (BISAG-N) for giving us this opportunity to work the guidance of renowned people of the field of MIS Based Portal also providing us with the required resources in the company.

We would like to express our endless thanks to our external guide **Mr. Vishal Patel**, And Admin Department **Mr. Sidhdharth Patel** at Bhaskaracharya National Institute of Space Application and Geo-informatics for their sincere and dedicated guidance throughout the project development.

Also, our hearty gratitude to our Head of Department, **Dr. Darshak Thakore** and our internal guide **Prof. Mayur Vegad** for giving us encouragement and technical support on the project.

Tanmay A Vaidya.
Student ID: H4

Deepnil K Vasava.
Student ID: H4

Deep J Patel.
Student ID: H4

CONTENTS

1. Introduction:

➢ Project Definition:

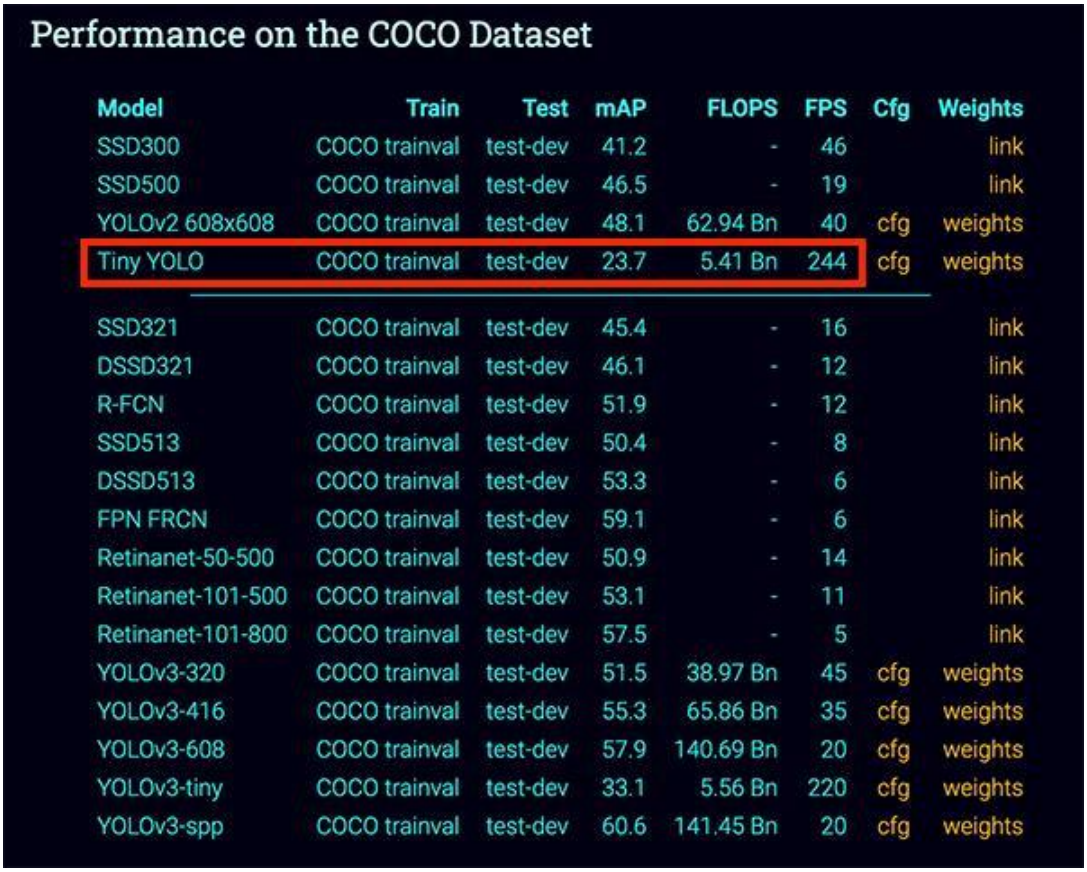
- To implement various **object detection algorithms** through **computer vision** and **neural network** models,
- Comparing accuracy, framerate, performance,
- Determining which algorithm works best for which video type (static/dynamic camera) or images.

[Image 1.1 Sample Object Detection]

- In the end the main purpose of this project is not to convert a particular object detection algorithm into a full-fledged software but to compare results of 5-6 different algorithms and determine their best usages.

➢ Why the need of Comparison?

- There are many applications of Object Detection which include:
- Car number plate detection for keeping traffic rules,
- Pedestrian Detection and Recognition (for advanced purposes),
- Facial Recognition,
- Emotion Recognition,
- Lane/ Traffic detection,
- All of these and many more are mainly used in security purposes and sometimes cover very sensitive matters and hence the algorithms/ models/ projects in general need to be as accurate and fast as possible.



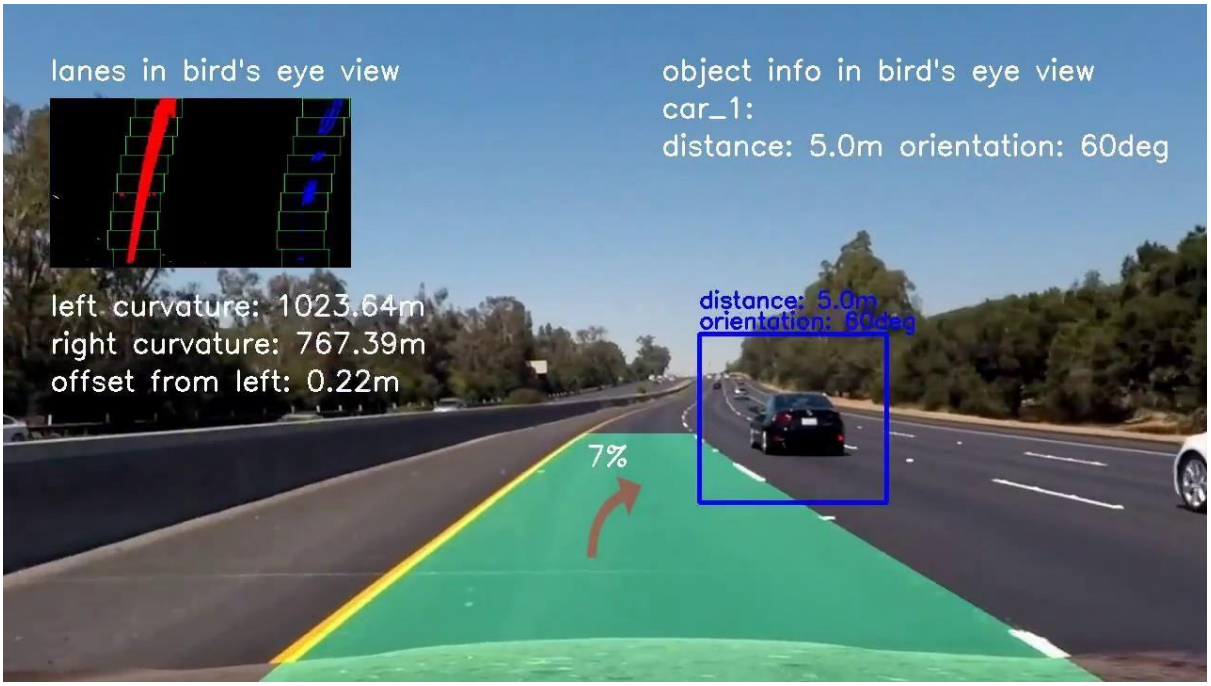
Performance on the COCO Dataset

Model	Train	Test	mAP	FLOPS	FPS	Cfg	Weights
SSD300	COCO trainval	test-dev	41.2	-	46		link
SSD500	COCO trainval	test-dev	46.5	-	19		link
YOLOv2 608x608	COCO trainval	test-dev	48.1	62.94 Bn	40	cfg	weights
Tiny YOLO	COCO trainval	test-dev	23.7	5.41 Bn	244	cfg	weights
SSD321	COCO trainval	test-dev	45.4	-	16		link
DSSD321	COCO trainval	test-dev	46.1	-	12		link
R-FCN	COCO trainval	test-dev	51.9	-	12		link
SSD513	COCO trainval	test-dev	50.4	-	8		link
DSSD513	COCO trainval	test-dev	53.3	-	6		link
FPN FRCN	COCO trainval	test-dev	59.1	-	6		link
Retinanet-50-500	COCO trainval	test-dev	50.9	-	14		link
Retinanet-101-500	COCO trainval	test-dev	53.1	-	11		link
Retinanet-101-800	COCO trainval	test-dev	57.5	-	5		link
YOLOv3-320	COCO trainval	test-dev	51.5	38.97 Bn	45	cfg	weights
YOLOv3-416	COCO trainval	test-dev	55.3	65.86 Bn	35	cfg	weights
YOLOv3-608	COCO trainval	test-dev	57.9	140.69 Bn	20	cfg	weights
YOLOv3-tiny	COCO trainval	test-dev	33.1	5.56 Bn	220	cfg	weights
YOLOv3-spp	COCO trainval	test-dev	60.6	141.45 Bn	20	cfg	weights

[Image 1.2 Sample Comparison of various algorithms on COCO dataset]

➢ Project Scope:

- An Object detection model can be combined with various **Hardware/ IOT combinations** like,
- Combining with CCTV camera for anti-theft systems in homes/ shopping malls/ ATMs etc.,
- Combining with automated cars for Lane/ Traffic/ Pedestrian detection/ tracking/ following etc.,
- Combining with Arduino/ raspberry Pi based IOT systems for object tracking/ detection / Obstacle avoidance/ Monitoring systems etc.,
- Combining with traffic cameras for number plate detection/ identification for breaking of traffic rules



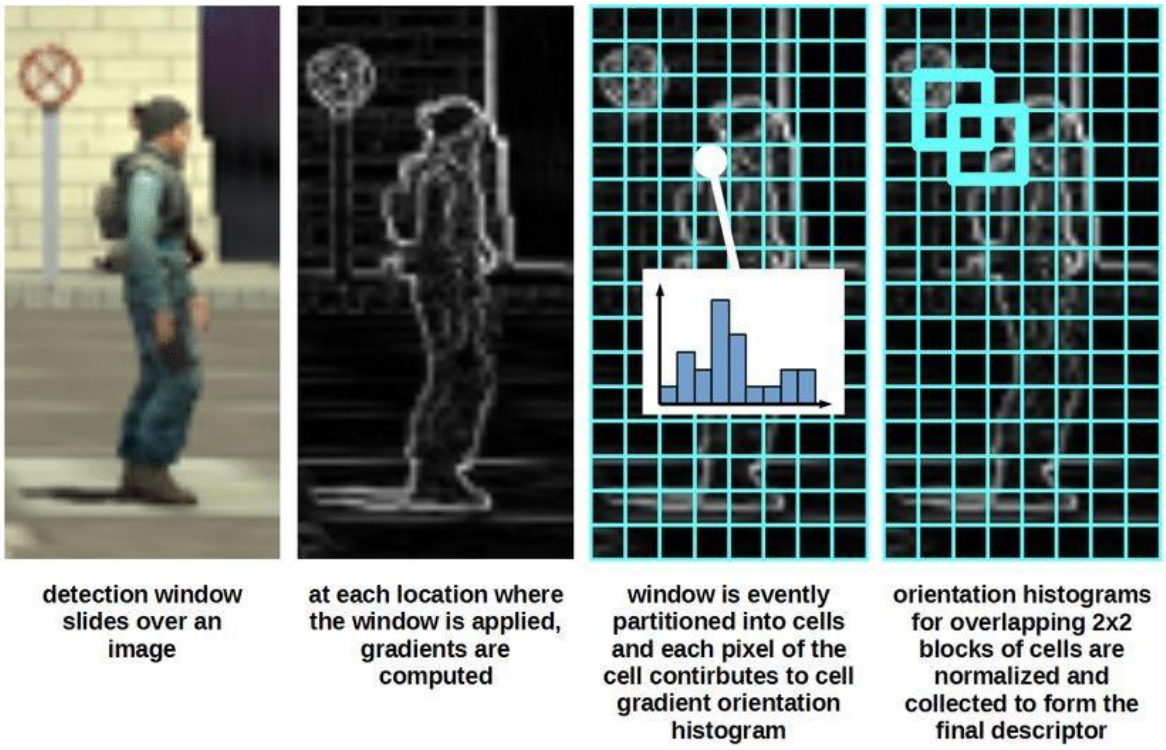
[Image 1.3 Sample Lane/ Traffic Tracking in cars]

- As mentioned, in the AI influenced future we're heading towards, Object Detection based models' scopes will only grow further with algorithms, cameras and machine learning libraries getting more and more advanced day by day.

➢ Brief Exploration of the algorithms used:

1. Histogram of Oriented Gradients:

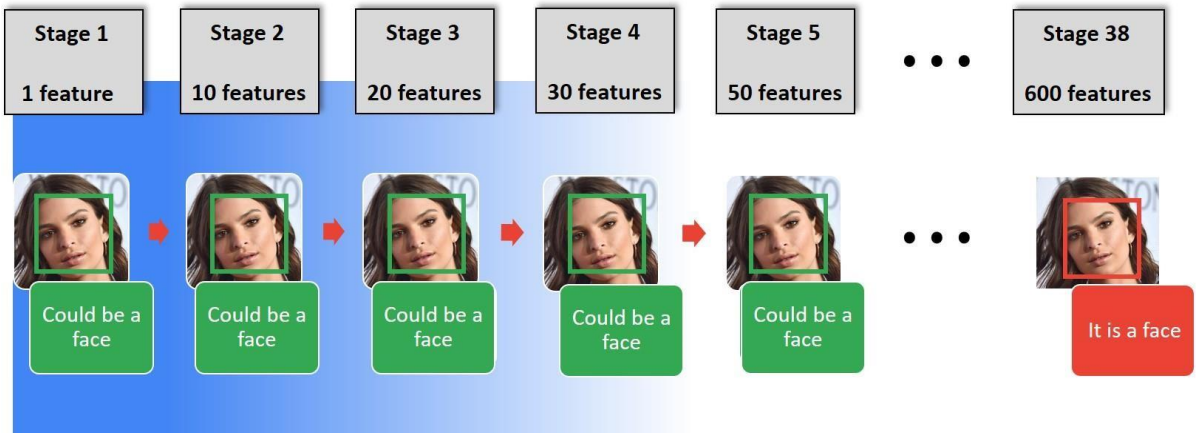
- Pretty basic algorithm in terms of providing accuracy and framerate in results,
- In the HOG feature descriptor, the distribution (histograms) of directions of gradients (oriented gradients) are used as features.
- Uses **64 x 128-pixel** sized images for required **8 x 8** and **16 x 16** matrices.
- **Gradients** (x and y derivatives) of an image are useful because the **magnitude of gradients** is large around edges and corners (regions of abrupt intensity changes) and we know that edges and corners pack in a lot more information about object shape than flat regions.



[Image 1.4 HOG step by step]

2. HAAR Cascade Classifier in computer vision:

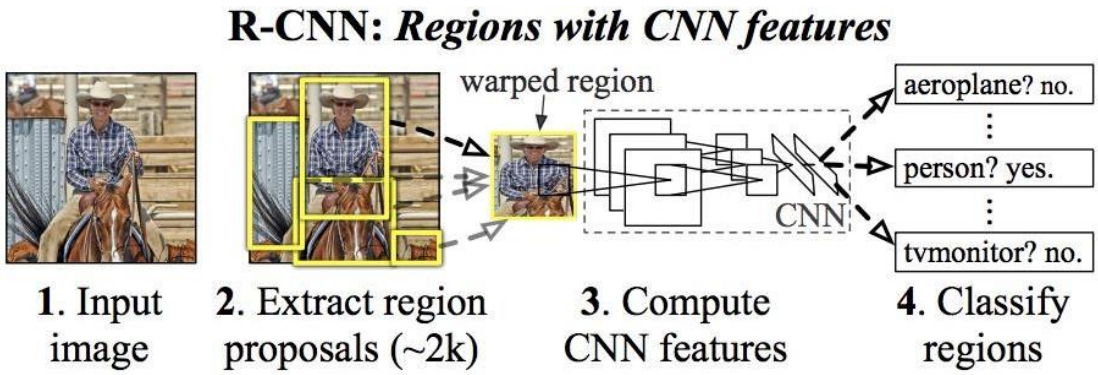
- Mostly included in Computer vision libraries like **OpenCV** (One we used), HAAR cascade classifier is a basic level Convolutional Neural Network Classifier (CNN).
- Mostly used for **facial recognition/ detection.**
- Accuracy depends on the dataset/ feature classifier used.
- Framerate depends in System/ GPU.



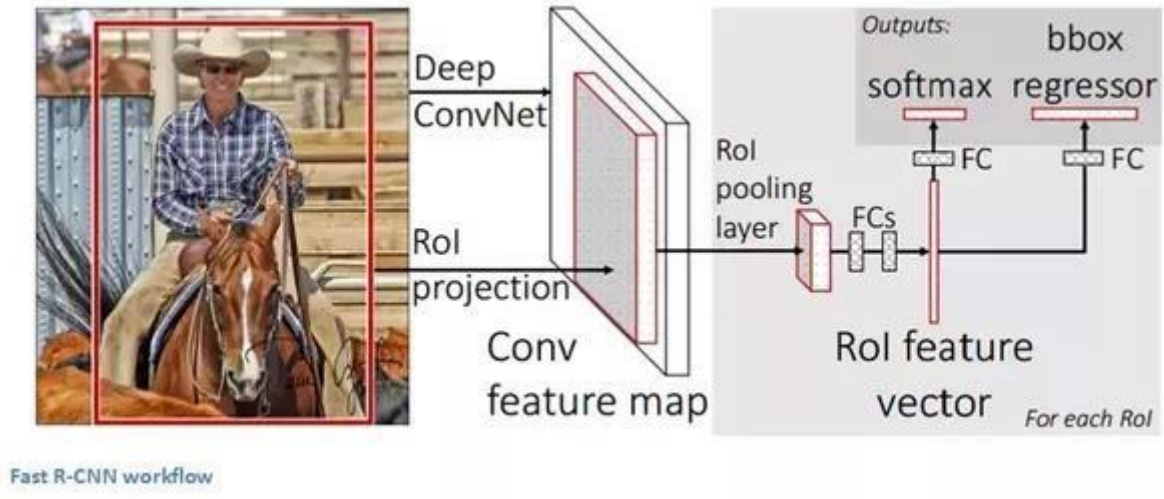
[Image 1.5 HAAR Cascade step by step]

3. RCNN and its evolution to Mask RCNN:

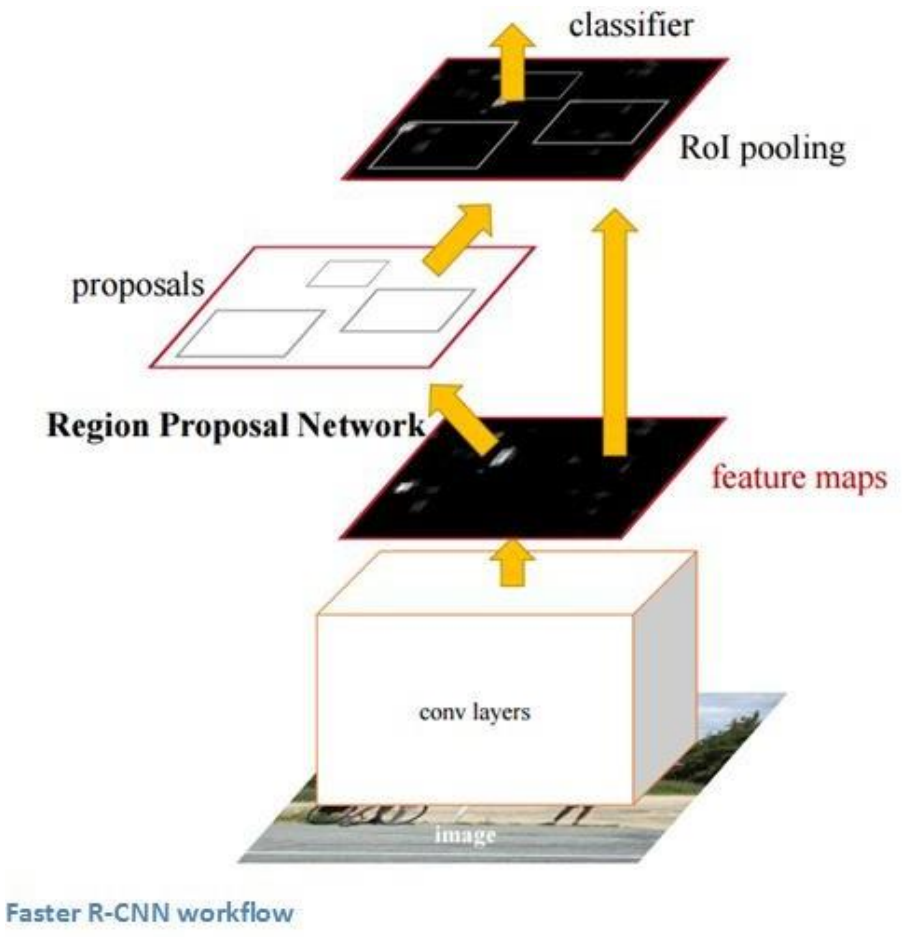
- **RCNN:** Independent of Region Proposals, it's basically a large CNN that extracts a fixed-length feature vector from each region.
- Mostly used with **Linear SVMs**, it produces decent results depending on the datasets and hardware used.
- Below is the process of RCNN and its evolution shown by images, be it Region Proposal networks, ConvNet, Masking RCNN is one of the most reliable when it comes to Object detection.



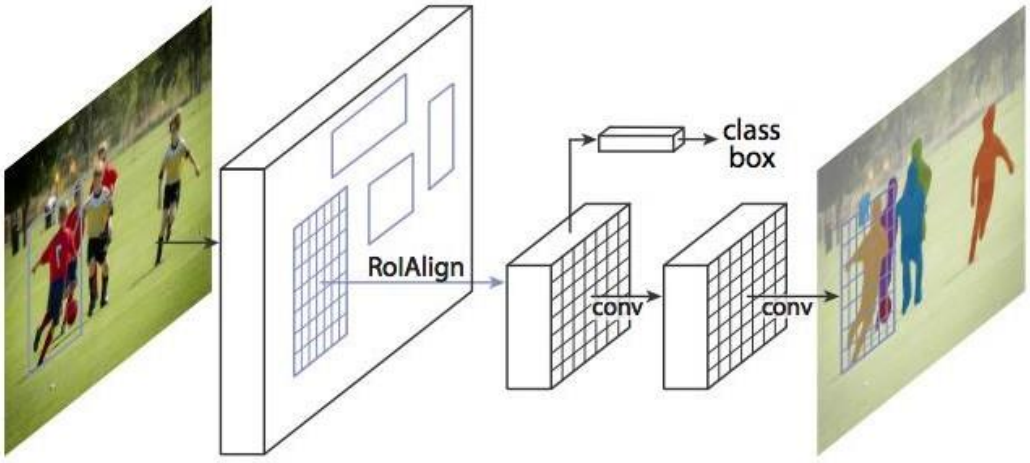
[Image 1.6 RCNN]



[Image 1.7 Fast RCNN]



[Image 1.8 Faster RCNN]

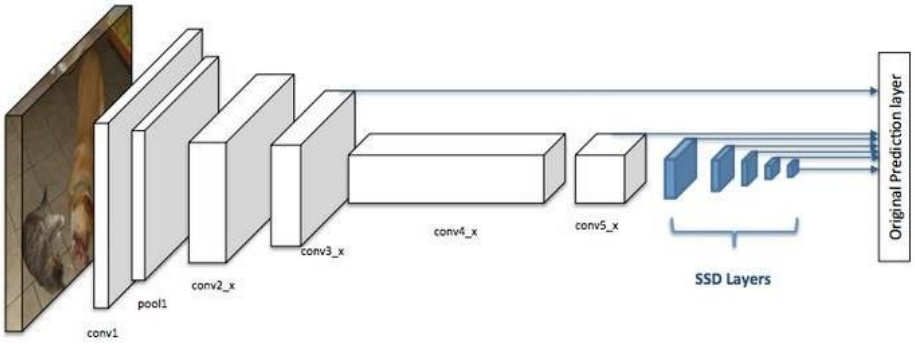


[Image 1.9 Mask RCNN]

- The One we used in this project is **Mask RCNN** as it's the most updated version as of now that RCNN based models have to offer.

4. Single Shot Detectors:

- SSD has two components: a **backbone** model and **SSD head**.
- ***Backbone*** model usually is a pre-trained image classification network as a feature extractor.
- This is typically a network like ResNet trained on ImageNet from which the final fully connected classification layer has been removed. We are thus left with a deep neural network that is able to extract semantic meaning from the input image while preserving the spatial structure of the image albeit at a lower resolution.
- For ResNet34, the backbone results in a 256 7x7 feature maps for an input image. We will explain what feature and feature map are later on.
- The ***SSD head*** is just one or more convolutional layers added to this backbone and the outputs are interpreted as the bounding boxes and classes of objects in the spatial location of the final layers' activations.
- It is one of the more advanced models used for object detection.



[Image 1.9]

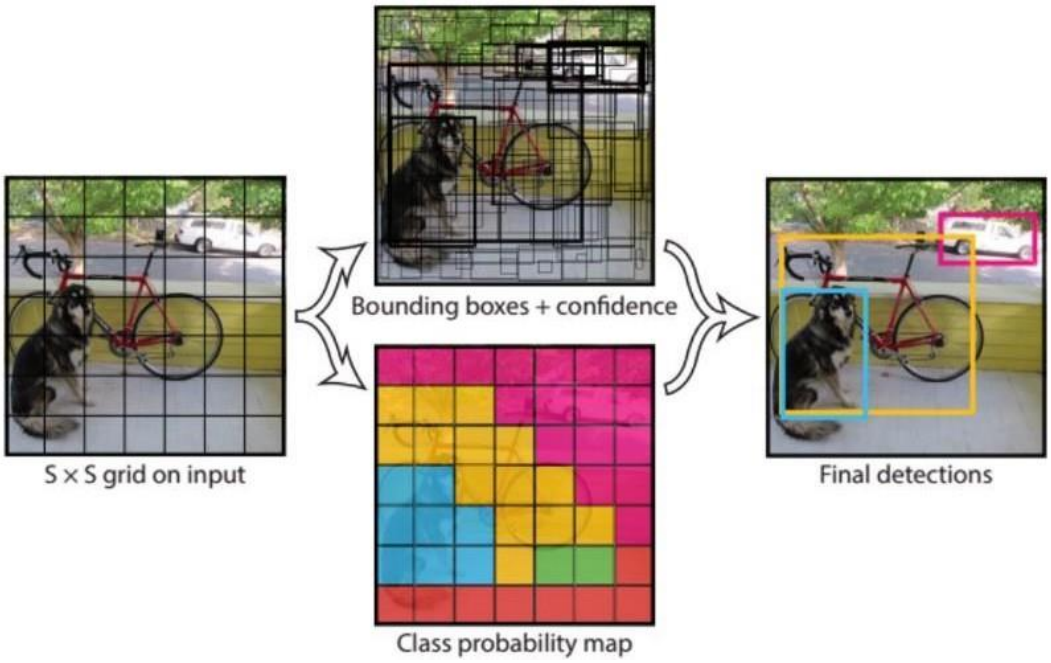
- In the figure above, the first few layers (white boxes) are the backbone, the last few layers (blue boxes) represent the SSD head.
- There are many SSD and model combinations available like with mobilenet, darknet, tensorflow etc.,

5. You Only Look Once:

- Like SSD, YOLO is one of the more advanced algorithms when it comes to object detection.

- Compared to other region proposal classification networks (fast RCNN) which perform detection on various region proposals and thus end up performing prediction multiple times for various regions in an image, **Yolo architecture is more like FCNN** (fully convolutional neural network) and passes the image (nxn) once through the FCNN and output is (mxm) prediction. This the architecture is splitting the input image in mxm grid and for each grid generation 2 bounding boxes and class probabilities for those bounding boxes. Note that bounding box is more likely to be larger than the grid itself. From paper (referenced):

- We reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities.

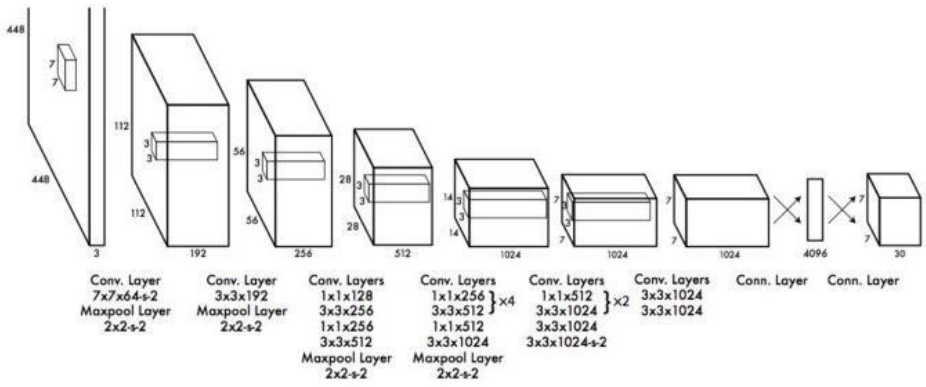


[Image 1.10]

- A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO

trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection. First, YOLO is extremely fast. Since we frame detection as a regression problem, we don't need a complex pipeline. We simply run our neural network on a new image at test time to predict detections. Our base network runs at 45 frames per second with no batch processing on a Titan X GPU and a fast version runs at more than 150 fps. This means we can process streaming video in real-time with less than 25 milliseconds of latency.

- **Fast YOLO** uses a neural network with fewer convolutional layers (9 instead of 24) and fewer filters in those layers. Other than the size of the network, all training and testing parameters are the same between YOLO and Fast YOLO.



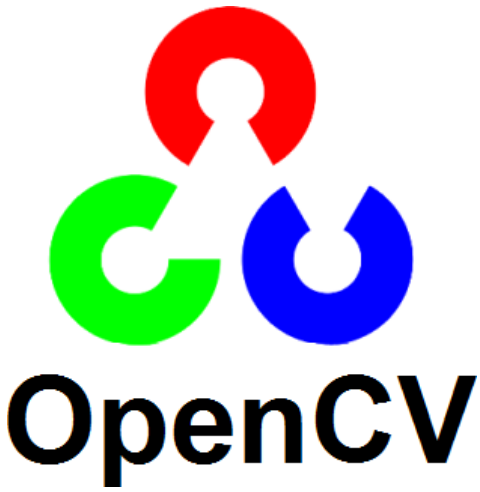
[Image 1.11]

Hence, we'll be comparing these 5 algorithms along with 2 approaches for SSD, we will be exploring 6 Object detection models.

2. Details of the Tools Used:

➢ Introduction to Open CV:

- OpenCV is a cross-platform library using which we can develop real-time **computer vision applications**. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.
- Computer Vision (also known as CV) can be defined as a discipline that explains how to reconstruct, interrupt, and understand a 3D scene from its 2D images, in terms of the properties of the structure present in the scene. It deals with modelling and replicating human vision using computer software and hardware
- **Image Processing, Pattern Recognition, Photogrammetry** are some fields that lie in Computer Vision.
- Here are some applications in various fields



[Image 2.1]

- **Robotics Application**
    - Localization − Determine robot location automatically
    - Navigation
    - Obstacle avoidance
    - Assembly (peg-in-hole, welding, painting)
    - Manipulation (e.g., PUMA robot manipulator)
    - Human Robot Interaction (HRI) − Intelligent robotics to interact with and serve people
- **Medicine Application**
    - Classification and detection (e.g., lesion or cells classification and tumour detection)
    - 2D/3D segmentation
    - 3D human organ reconstruction (MRI or ultrasound)
    - Vision-guided robotics surgery
- **Industrial Automation Application**
    - Industrial inspection (defect detection)
    - Assembly
    - Barcode and package label reading
    - Object sorting
    - Document understanding (e.g., OCR)
- **Security Application**
    - Biometrics (iris, finger print, face recognition)
    - Surveillance − Detecting certain suspicious activities or behaviours
- **Transportation Application**
    - Autonomous vehicle
    - Safety, e.g., driver vigilance monitoring

Features of OpenCV Library:

- Using OpenCV library, you can −
    - Read and write images
    - Capture and save videos

- Process images (filter, transform)
- Perform feature detection
- Detect specific objects such as faces, eyes, cars, in the videos or images.
- Analyse the video, i.e., estimate the motion in it, subtract the background, and track objects in it.

OpenCV was mostly created for C++ but later integrated with other languages like Python and Java.

➢ Open CV Modules:

- Core Functionality
- Image Processing
- Video
- Video I/O
- calib3d
- features2d
- Objdetect
- Highgui

Installation:

- Python: - pip install opencv-python
- Conda: - conda install -c conda-forge opencv

Things we can do with Open CV:

1. Image Conversion

- coloured images to grayscale

- convert grayscale images to binary image
- convert a coloured to a binary image.

2. Drawing Functions

- draw various shapes like Circle, Rectangle, Line, Ellipse, Polylines, Convex, Polylines.

3. Blur

- Blur (Averaging), Gaussian Blur, Median Blur.

4. Filtering

- filter operations such as Bilateral Filter, Box Filter, SQR Box Filter and Filter2D.

5. Thresholding

- simple thresholding and adaptive thresholding.

6. Transformation Operations

- Laplacian Transformation
- Distance Transformation

7. Camera and Face Detection

- Using Camera
- Face detection using camera

- Face detection using picture

8. Extras

- Canny edge detection
- Hough line transform
- Histogram equalization

➢ Introduction to TensorFlow

- TensorFlow is an open-source machine learning framework for all developers. It is used for implementing machine learning and deep learning applications. To develop and research on fascinating ideas on artificial intelligence, Google team created TensorFlow. TensorFlow is designed in Python programming language, hence it is considered an easy-to-understand framework.

[Image 2.2]

Some of the key features of TensorFlow:

- It includes a feature of that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.

- It includes a programming support of deep neural networks and machine learning techniques.
- It includes a high scalable feature of computation with various data sets.
- TensorFlow uses GPU computing, automating management. It also includes a unique feature of optimization of same memory and the data used.

Installation of TensorFlow

- pip install tensorflow
- pip install tensorflow-gpu

Note: - its recommended to install TensorFlow in Virtual Env as its solely depended upon the python version.

Machine Learning and Deep Learning

- **Machine learning** is the art of science of getting computers to act as per the algorithms designed and programmed
- **Deep learning** is a subfield of machine learning where concerned algorithms are inspired by the structure and function of the brain called artificial neural networks.

TensorFlow Keras

- Keras is compact, easy to learn, high-level Python library run on top of TensorFlow framework. It is made with focus of understanding deep learning techniques, such as creating layers for neural networks maintaining the concepts of shapes and mathematical details. The creation of framework can be of the following two types −

- Sequential API
- Functional API

Consider the following eight steps to create deep learning model in Keras −

1. Loading the data
2. Pre-processing
3. Definition of model
4. Compiling the model
5. Fit the specified model
6. Evaluate it
7. Make the required predictions
8. Save the model

TensorFlow includes a special feature of image recognition and these images are stored in a specific folder. With relatively same images, it will be easy to implement this logic for security purposes.

Pixellib:-

pixellib is a library for performing segmentation of objects in images and videos. It supports the two major types of image segmentation:

1. **Semantic segmentation**
2. **Instance segmentation**

PixelLib uses object segmentation to perform excellent foreground and background separation. It makes possible to alter the background of any image and video using just five lines of code.

The following features are supported for background editing,

1. **Create a virtual background for an image and a video**
2. **Assign a distinct color to the background of an image and a video**
3. **Blur the background of an image and a video**
4. **Grayscale the background of an image and a video**

➢ IDEs Used

1. Jupyter notebook: -

- The **Jupyter Notebook** App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.
- It is available through anaconda.

2. Pycharm: -

- **PyCharm** is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development.

3. Spyder: -

- **Spyder** is a powerful scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It offers a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package.

4. Colab: -

- **Colaboratory** is a Google research project created to help disseminate machine learning education and research. It's a Jupyter notebook environment that requires no setup to use and runs entirely in the cloud.

3. Project Analysis

➢ object detection introduction:

- Object detection is a combination of object classification as well as object localization which makes it one of the most challenging topics in the domain of computer vision. The goal of this detection technique is to determine where objects are located in a given image called as object localization and which category each object belongs to, that is called as object classification.
- Interpreting the object localization can be done in various ways, including creating a bounding box around the object or marking every pixel in the image which contains the object (called segmentation).

[Image 3.1 Object Detection using bounding boxes]

[Image 3.2 Object Segmentation by predicting pixel-level masks]

Object detection back in old days

- **Object detection before Deep Learning was a several step process, starting with edge detection and feature extraction using techniques like SIFT (scale-invariant feature transform), HOG (**Histogram of Oriented Gradients**) etc. These images were then compared with existing object templates, usually at multi scale levels, to detect and localize objects present in the image.**

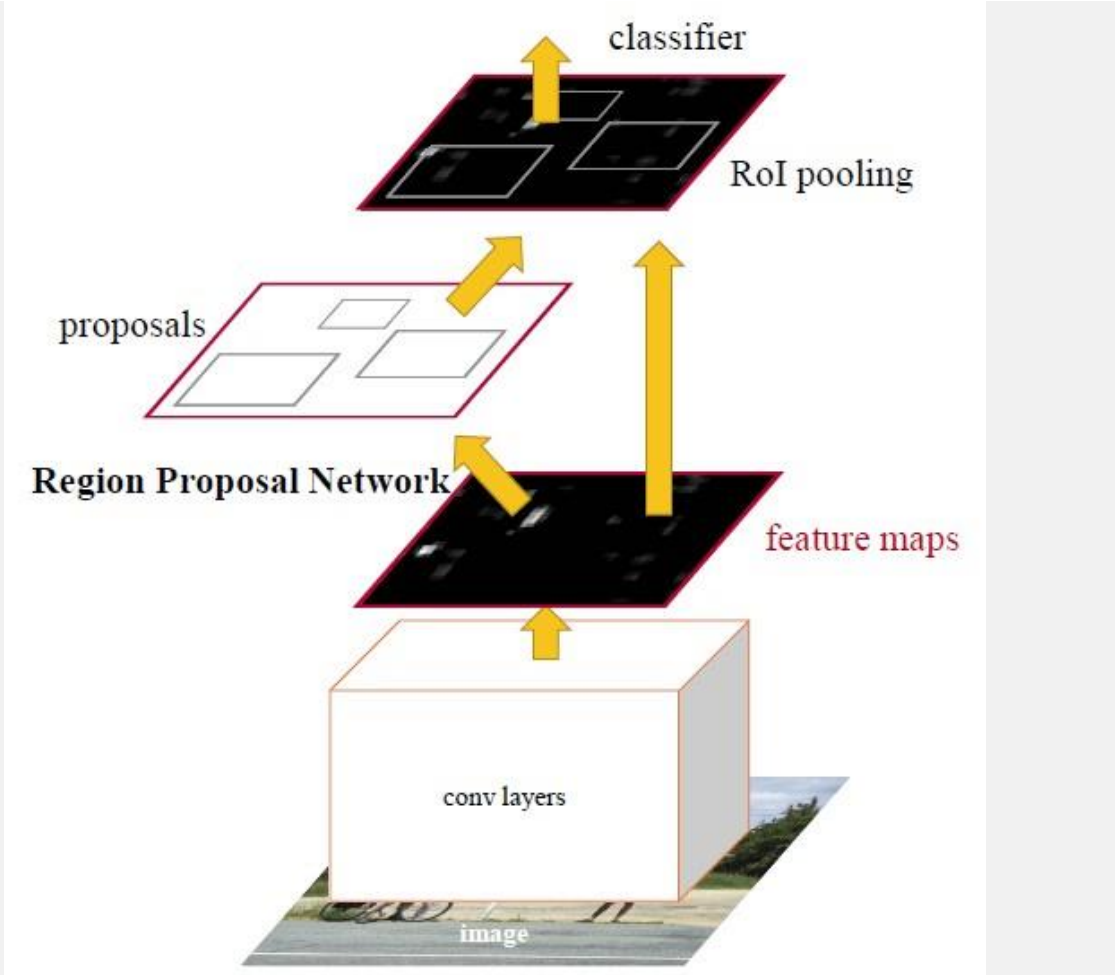
Object detection now

➢ **Two-Step Object Detection**

- Two-Step Object Detection involves algorithms that first identify bounding boxes which may potentially contain objects and then classify each bounding separately.

- The first step requires a Region Proposal Network, providing a number of regions which are then passed to common DL (Deep Learning) based classification architectures. From the hierarchical grouping algorithm in RCNNs (which are extremely slow) to using CNNs (Convolutional Neural Network) and ROI (Region of Interest) pooling in Fast RCNNs and anchors in Faster RCNNs (thus speeding up the pipeline and training end-to-end), a lot of different methods

and variations have been provided to these region proposal networks (RPNs).

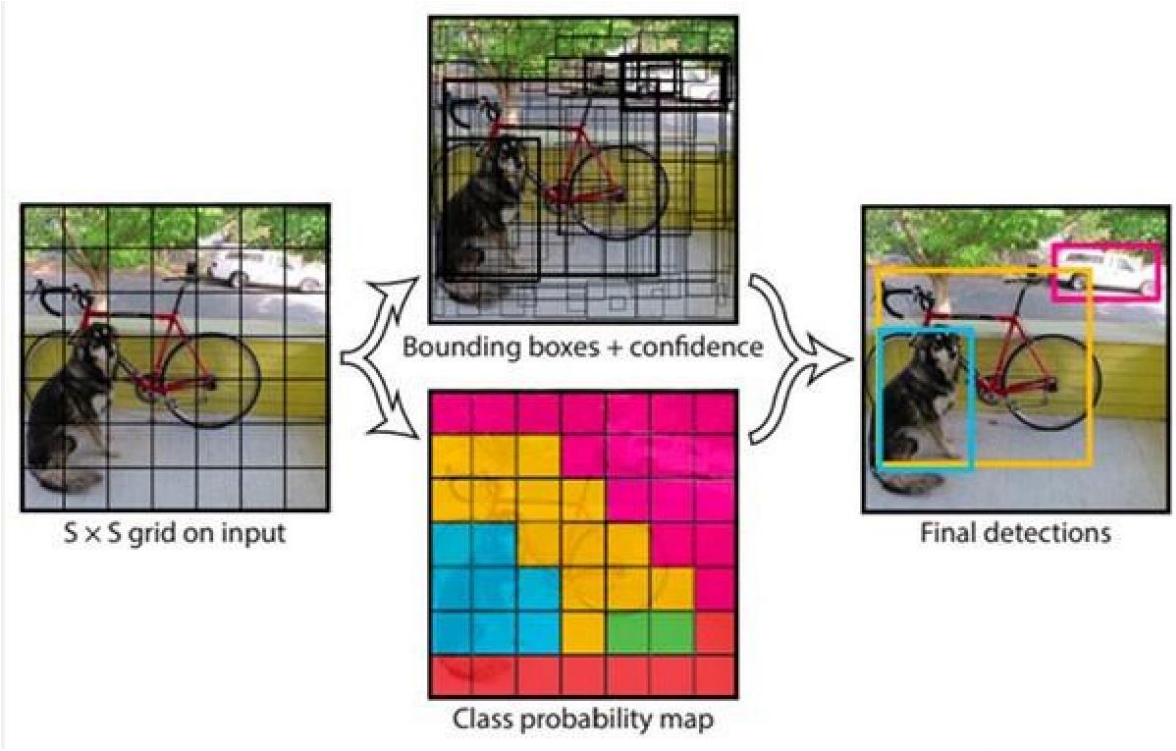


- These algorithms are known to perform better than their one-step object detection counterparts, but are slower in comparison.

➢ One-Step Object Detection

- With the need of real time object detection, many one-step object detection architectures have been proposed, like YOLO, YOLOv2, YOLOv3, SSD, RetinaNet etc. which try to combine the detection and classification step.

- One of the major accomplishments of these algorithms have been introducing the idea of ‘regressing’ the bounding box predictions. When every bounding box is represented easily with a few values (for example, xmin, xmax, ymin and ymax), it becomes easier to combine the detection and classification step and dramatically speed up the pipeline.



- For example, YOLO divided the entire image into smaller grid boxes. For each grid cell, it predicts the class probabilities and the x and y coordinates of every bounding box which passes through that grid cell. Kind of like the image-based captcha where you select all smaller grids which contain the object.

- These modifications allow one-step detectors to run faster and also work on a global level. However, since they do not work on every bounding box separately, this can cause them to perform worse in case of smaller objects or similar object in close vicinity. There have been multiple new architectures introduced to give more importance to lower-level features too, thus trying to provide a balance.

Basic Object detection algorithms

1. RCNN

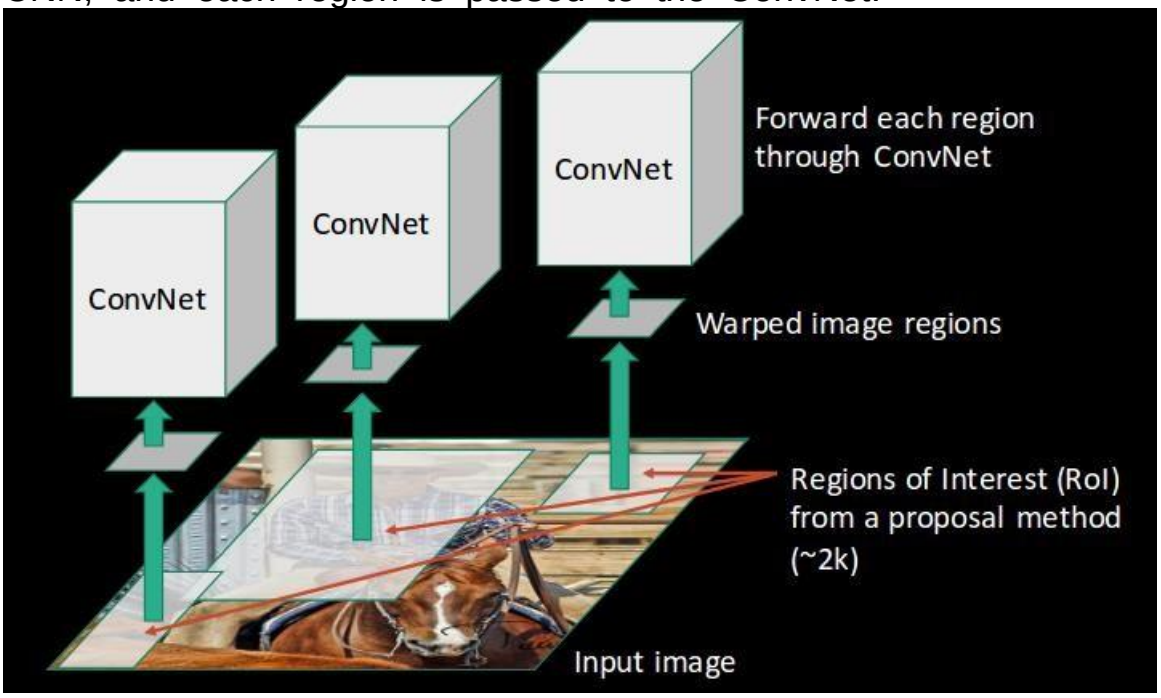
Instead of working on a massive number of regions, the RCNN algorithm proposes a bunch of boxes in the image and checks if any of these boxes contain any object. RCNN **uses selective search to extract these boxes from an image (these boxes are called regions).**

The steps followed in RCNN to detect objects are as follows: -

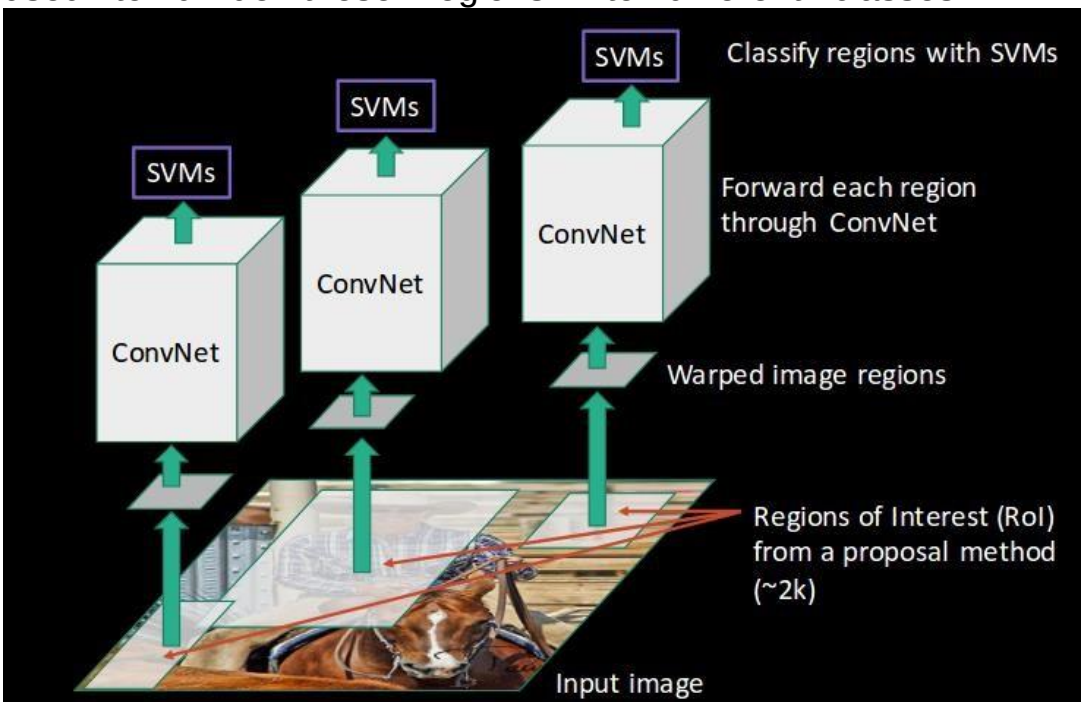
- First an image is taken as an input:

- Then, we get the Regions of Interest (ROI) using some proposal method (for example, selective search as seen above):

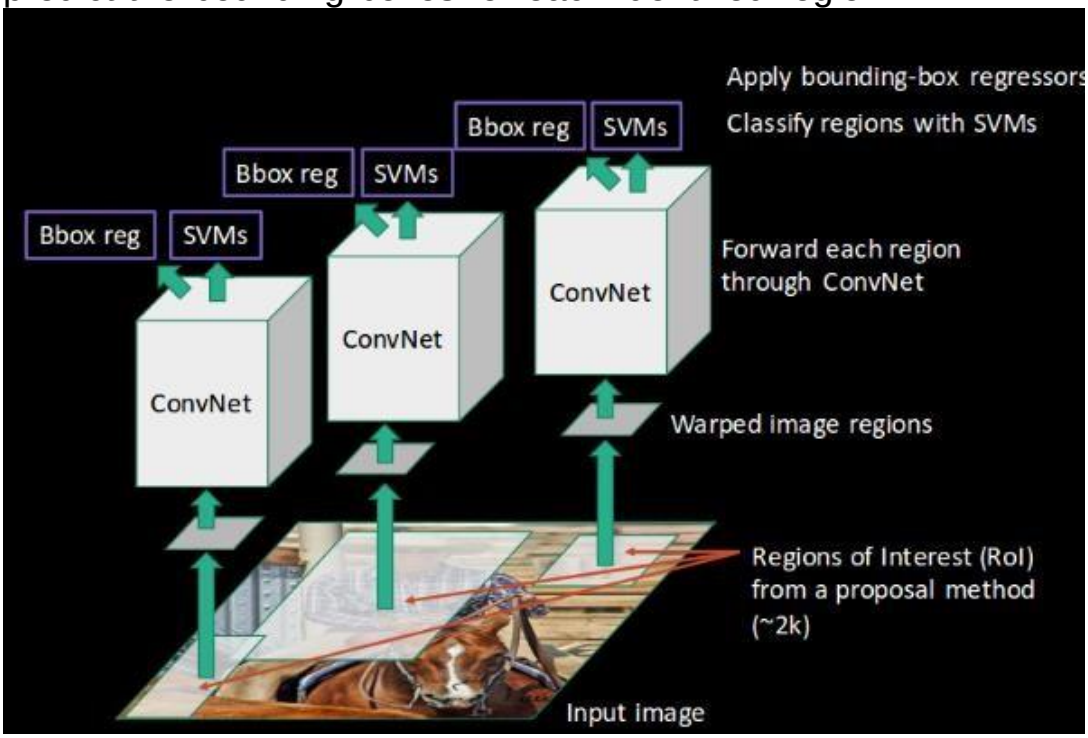
- All these regions are then reshaped as per the input of the CNN, and each region is passed to the ConvNet:



- CNN then extracts features for each region and SVMs are used to divide these regions into different classes:



- Finally, a bounding box regression (*Bbox reg*) is used to predict the bounding boxes for each identified region:



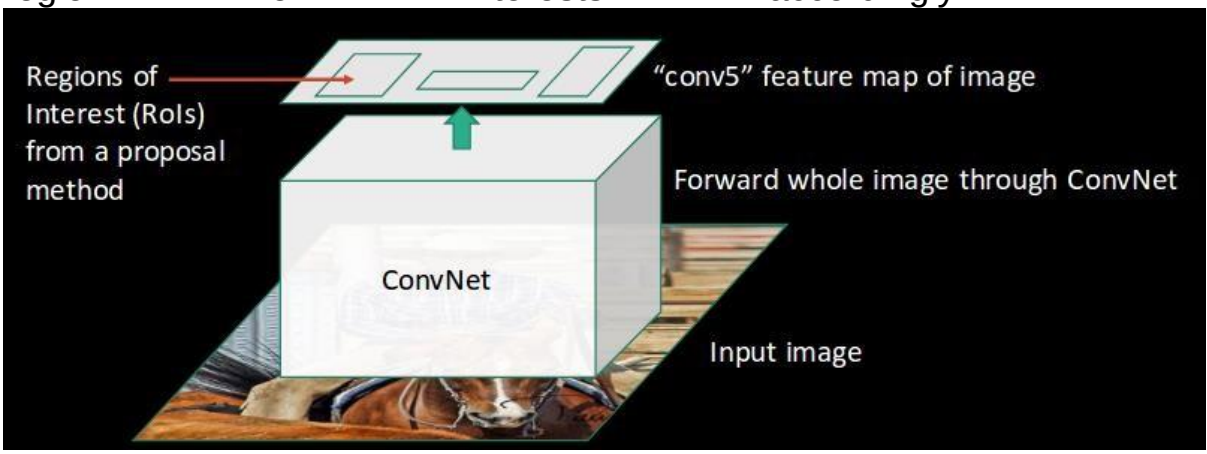
2. Fast RCNN

Instead of running a CNN 2,000 times per image, we can run it just once per image and get all the regions of interest (regions containing some object).

The steps followed in RCNN to detect objects are as follows: -

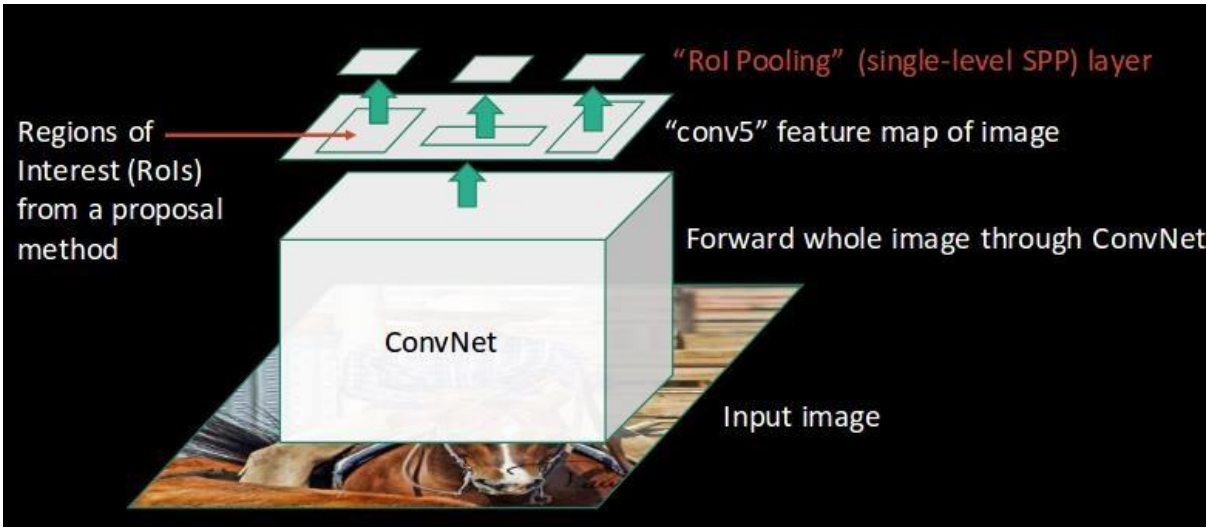
- We follow the now well-known step of taking an image as input:

- This image is passed to a ConvNet which returns the region of interests accordingly:

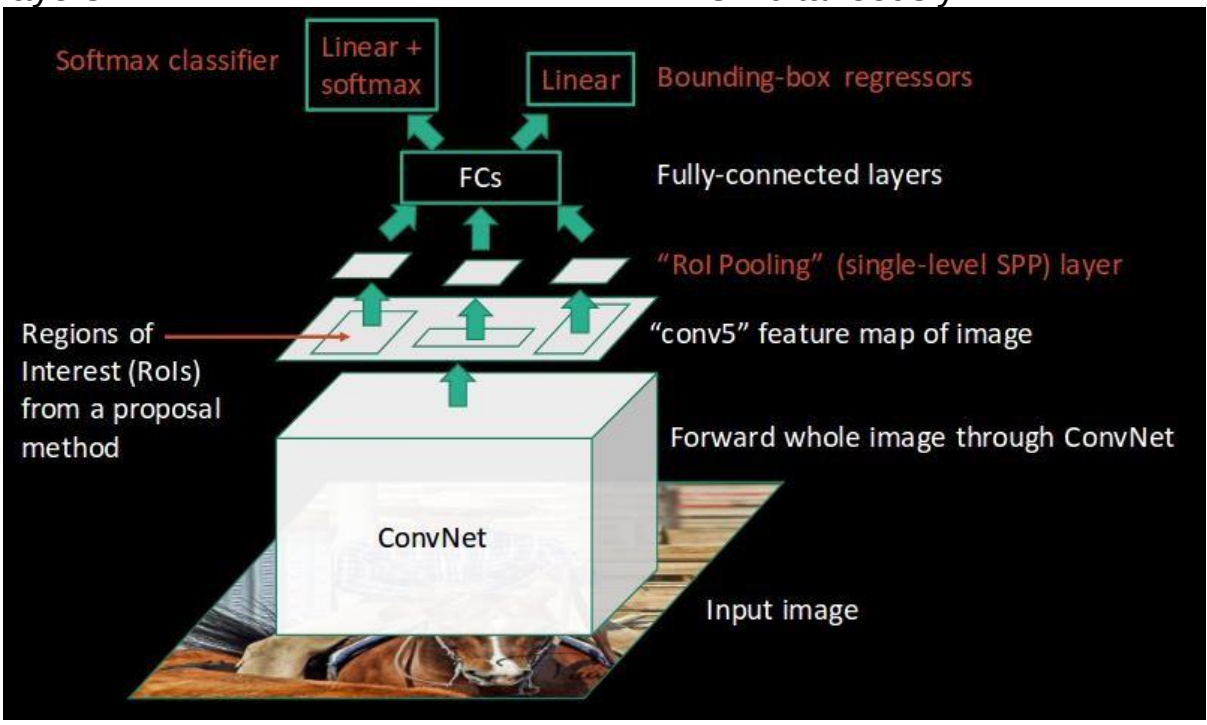


- Then we apply the RoI pooling layer on the extracted regions of interest to make sure all the regions are of the

same size:



- Finally, these regions are passed on to a fully connected network which classifies them, as well as returns the bounding boxes using SoftMax and linear regression layers simultaneously:



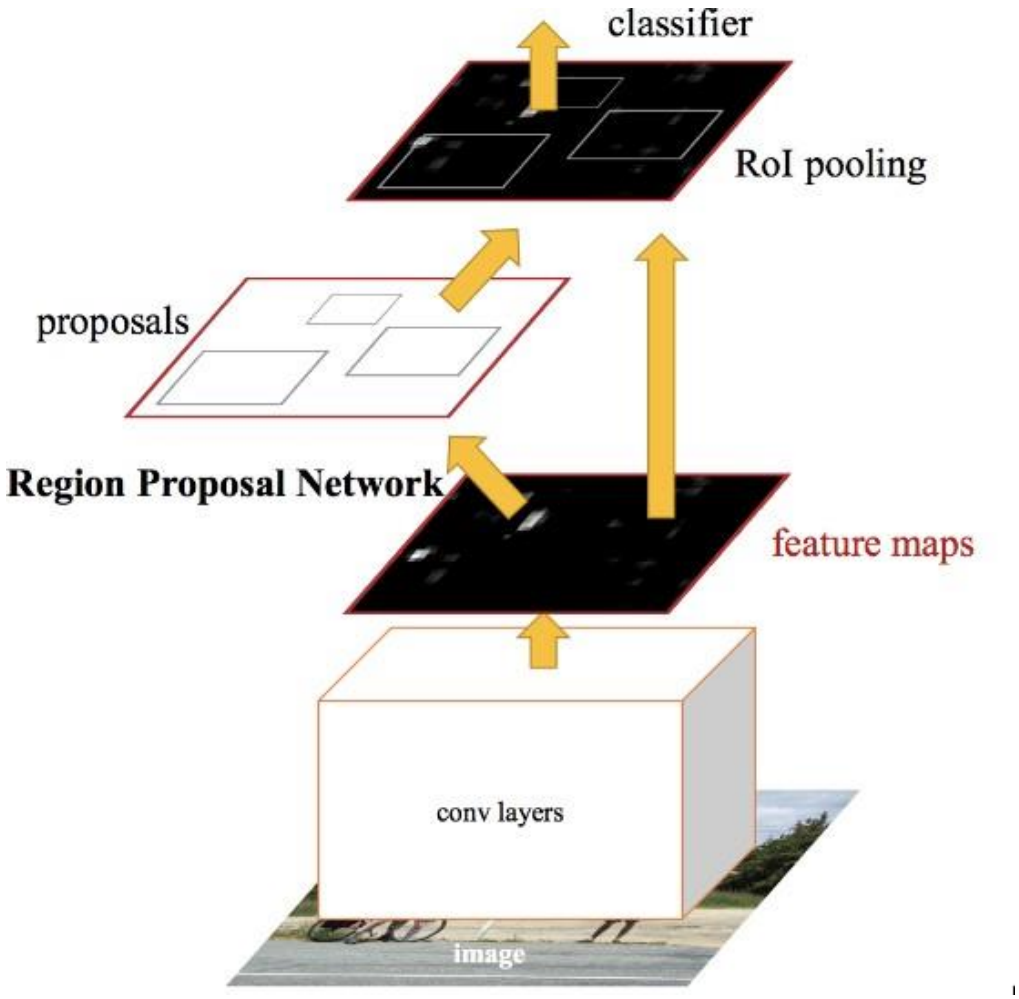
This is how Fast RCNN resolves two major issues of RCNN, i.e., passing one instead of 2,000 regions per image to the ConvNet, and using one instead of three different models for extracting features, classification and generating bounding boxes.

3. Faster RCNN

Faster RCNN is the modified version of Fast RCNN. The major difference between them is that Fast RCNN uses selective search for generating Regions of Interest, while Faster RCNN uses "Region Proposal Network", aka RPN. RPN takes image feature maps as an input and generates a set of object proposals, each with an objectless score as output.

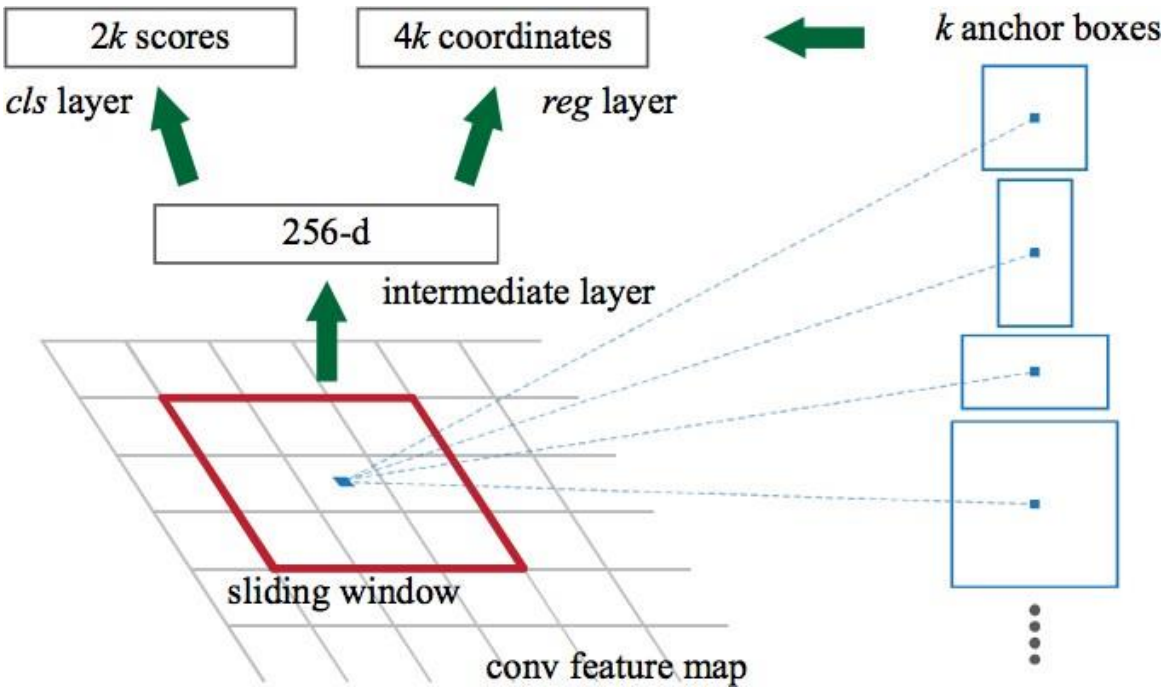
The below steps are typically followed in a Faster RCNN approach:

1. We take an image as input and pass it to the ConvNet which returns the feature map for that image.
2. Region proposal network is applied on these feature maps. This returns the object proposals along with their objectless score.
3. A RoI pooling layer is applied on these proposals to bring down all the proposals to the same size.
4. Finally, the proposals are passed to a fully connected layer which has a softmax layer and a linear regression layer at its top, to classify and output the bounding boxes for objects.



Let me briefly explain how this Region Proposal Network (RPN) actually works.

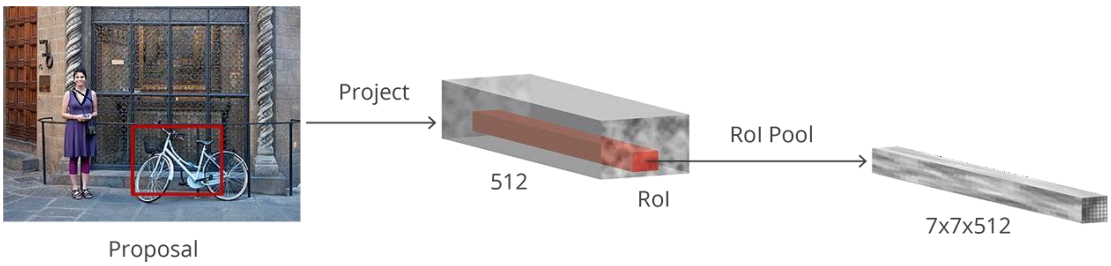
To begin with, Faster RCNN takes the feature maps from CNN and passes them on to the Region Proposal Network. RPN uses a sliding window over these feature maps, and at each window, it generates *k* Anchor boxes of different shapes and sizes:



Anchor boxes are fixed sized boundary boxes that are placed throughout the image and have different shapes and sizes. For each anchor, RPN predicts two things:

- The first is the probability that an anchor is an object (it does not consider which class the object belongs to)
- Second is the bounding box regressor for adjusting the anchors to better fit the object

We now have bounding boxes of different shapes and sizes which are passed on to the RoI pooling layer. Now it might be possible that after the RPN step, there are proposals with no classes assigned to them. We can take each proposal and crop it so that each proposal contains an object. This is what the RoI pooling layer does. It extracts fixed sized feature maps for each anchor:



Then these feature maps are passed to a fully connected layer which has a softmax and a linear regression layer. It finally classifies the object and predicts the bounding boxes for the identified objects.

Problem Identification:

1. Problems with RCNN

So far, we've seen how RCNN can be helpful for object detection. But this technique comes with its own limitations. Training an RCNN model is expensive and slow thanks to the below steps:

- Extracting 2,000 regions for each image based on selective search
- Extracting features using CNN for every image region. Suppose we have N images, then the number of CNN features will be N*2,000
- The entire process of object detection using RCNN has three models:
    1. CNN for feature extraction
    2. Linear SVM classifier for identifying objects
    3. Regression model for tightening the bounding boxes.

All these processes combine to make RCNN very slow. It takes around 40-50 seconds to make predictions for each new image, which essentially makes the model cumbersome and practically impossible to build when faced with a gigantic dataset.

2. Problems with Fast RCNN

Even Fast RCNN has certain problem areas. It also uses selective search as a proposal method to find the Regions of Interest, which is a slow and time-consuming process. It takes around 2 seconds per image to detect objects, which is much better compared to RCNN. But when we consider large real-life datasets, then even a Fast RCNN doesn't look so fast anymore.

3. Problems with Faster RCNN

The network does not look at the complete image in one go, but focuses on parts of the image sequentially. This creates two complications:

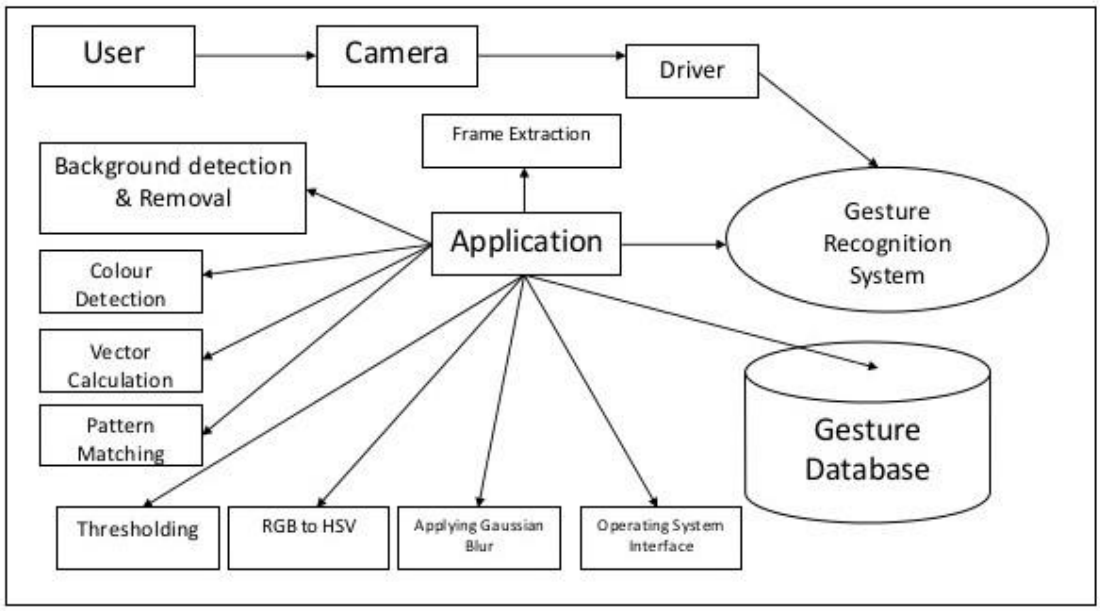
- The algorithm requires many passes through a single image to extract all the objects
- As there are different systems working one after the other, the performance of the systems further ahead depends on how the previous systems performed

Feasibility:

Technical-Feasibility: As far as machine learning (even more so in neural networks) is concerned a high processing system is required for maximum accuracy.

4. Designing aspects:

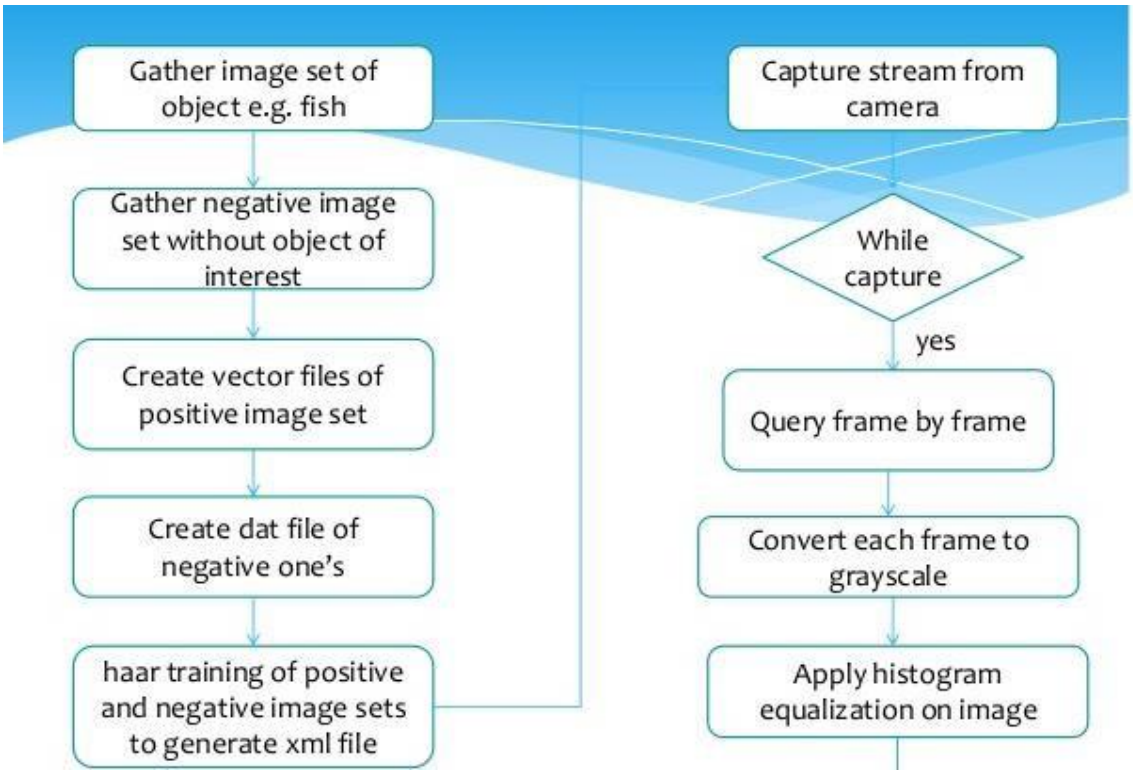
- Here is a sample DFD for Object detection via Neural Networks:



[4.1 Gesture Recognition DFD]

It's a fairly simple DFD based on gesture detection, we have applied the algorithms in different ways like pedestrian/ face detection and this diagram mostly represents the steps we've taken.

- Here is a sample Flow chart for HAAR Cascade classifier:



As mentioned, first we perform Pre-Processing on the input images like,

- Reshaping/ Resizing,
- Masking,
- Converting to suitable format

We proceed with the dataset or database given and train our input file,

Then appropriate detection algorithm is applied,

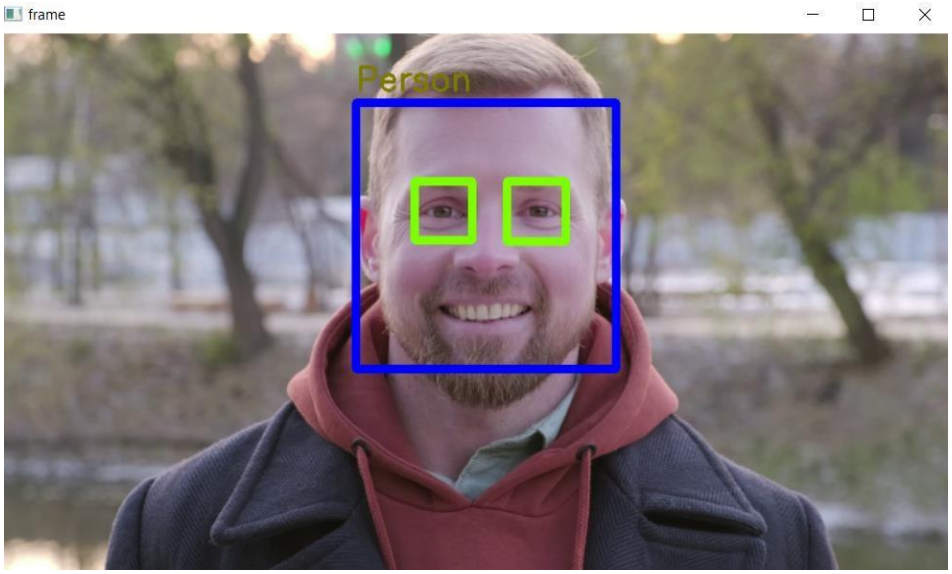
Moving on, we analyze the results and compare.

We opted not to create separate different diagrams for all 6 models we used as this is mainly a research-based project and the sample diagrams show our approaches fairly well.

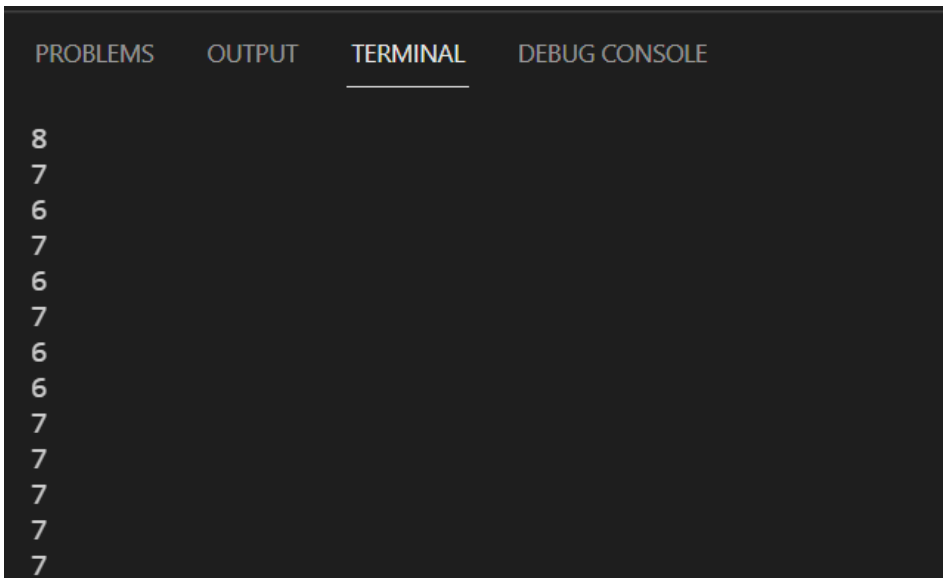
5. Screenshots

HOG Classifier:

Detection:



Framerate:



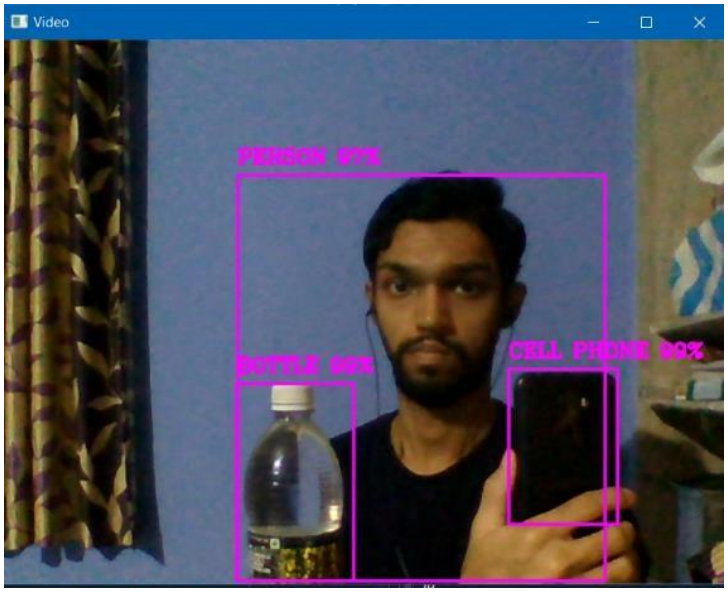
HAAR Cascade:

Detection:

Framerate:

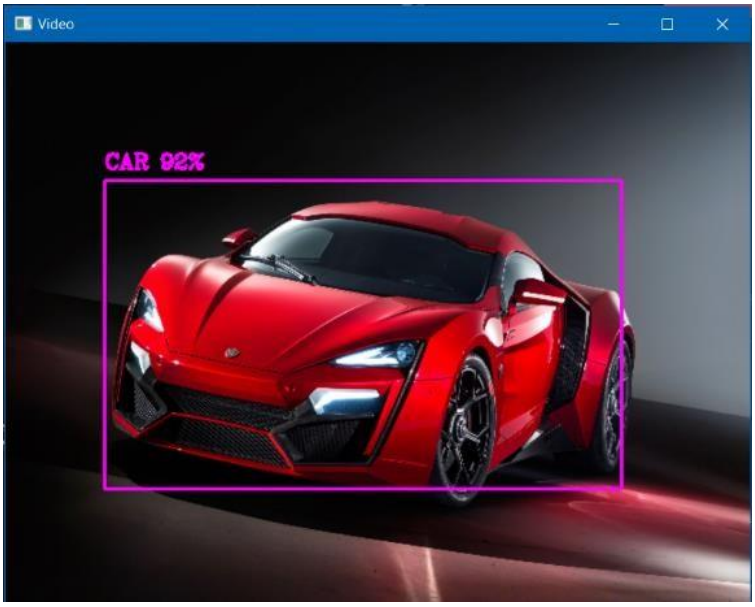
YOLO (You Only Look Once) Object Detection - v3

Webcam:



Video:

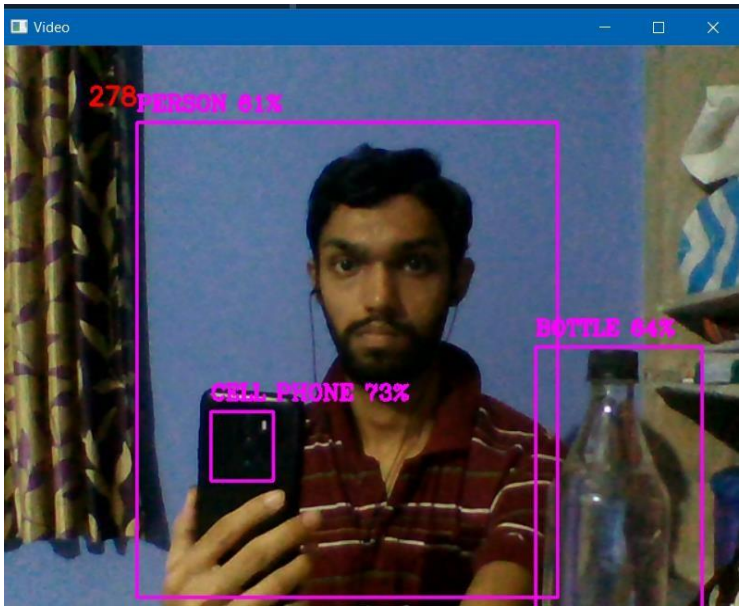
Image:



YOLO

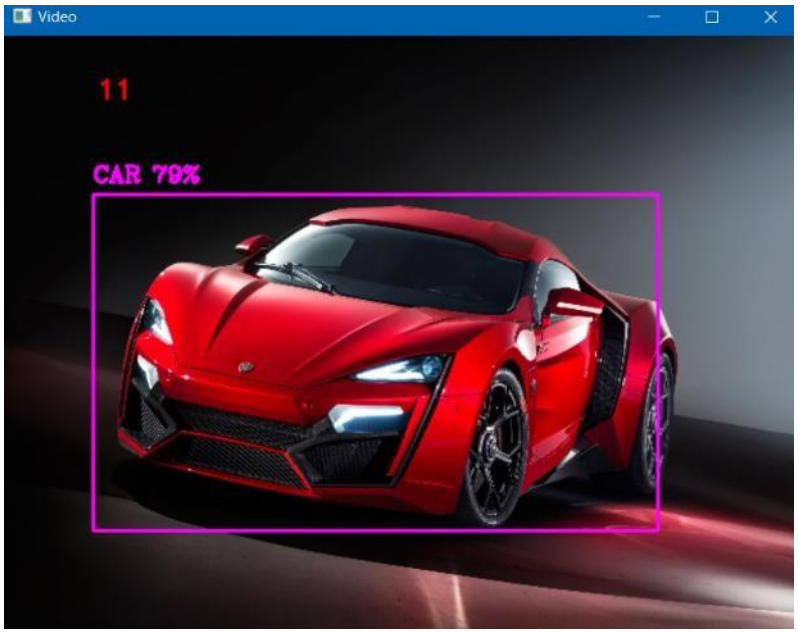
Object Detection - tiny

Webcam:



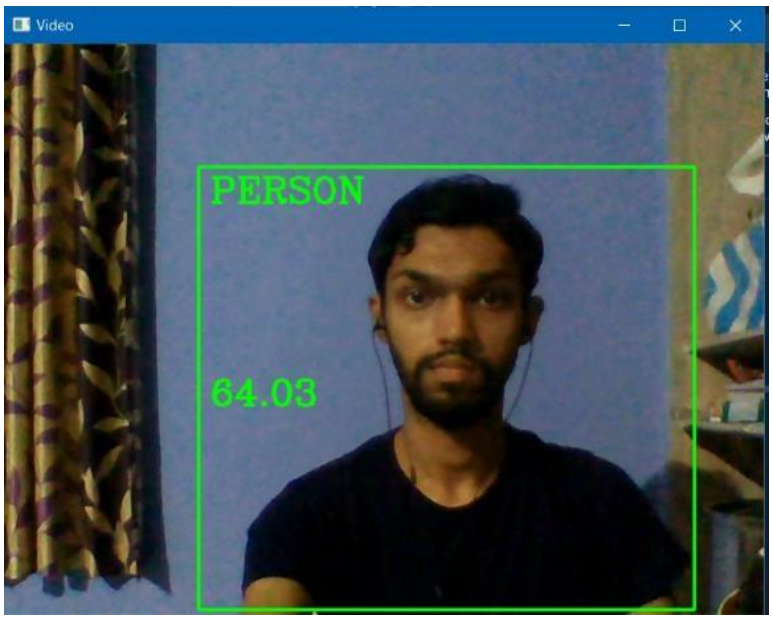
Video:

Image:



SSD:

Webcam:



Video:

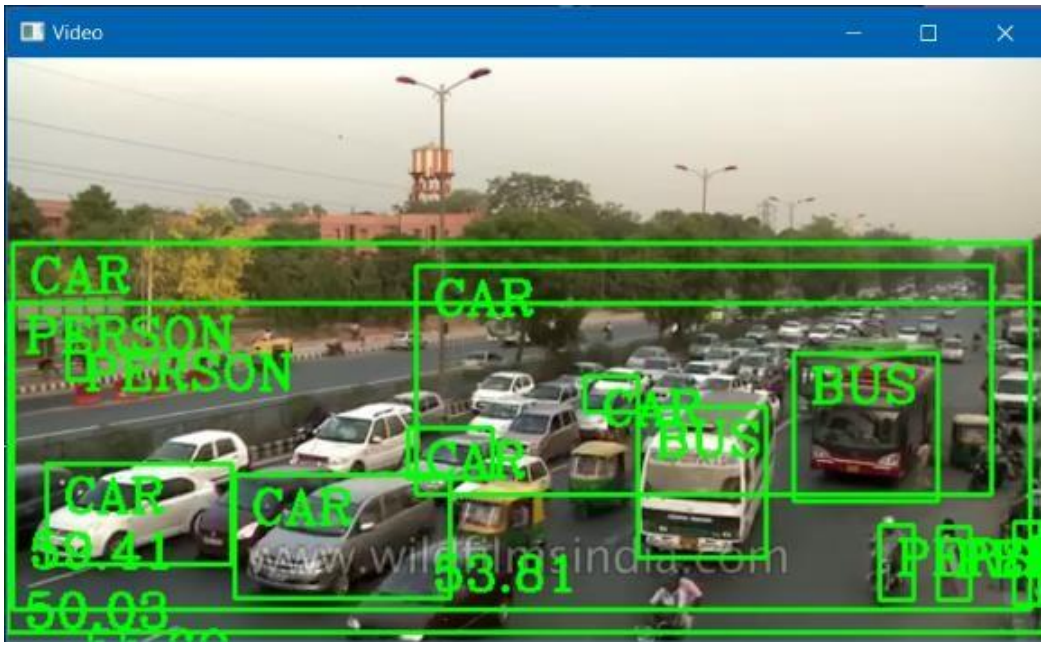
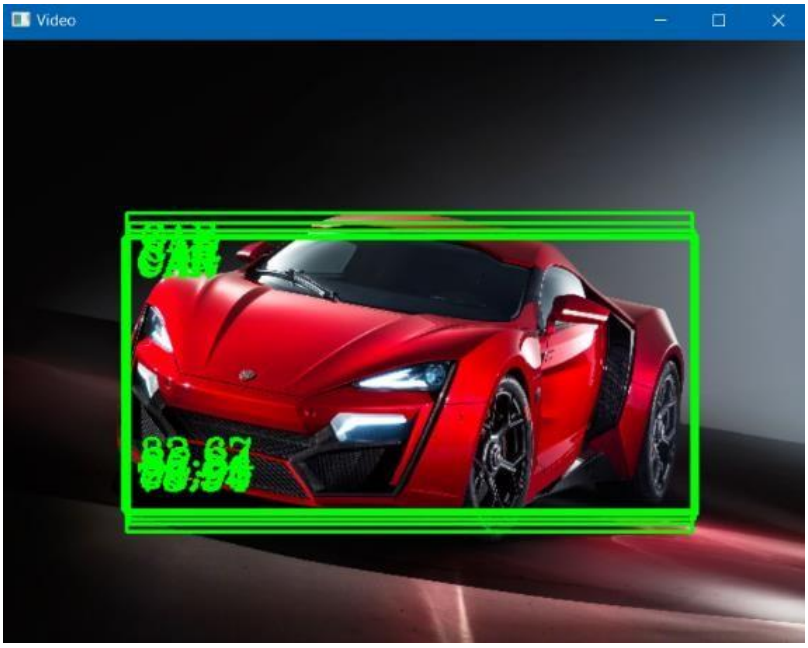
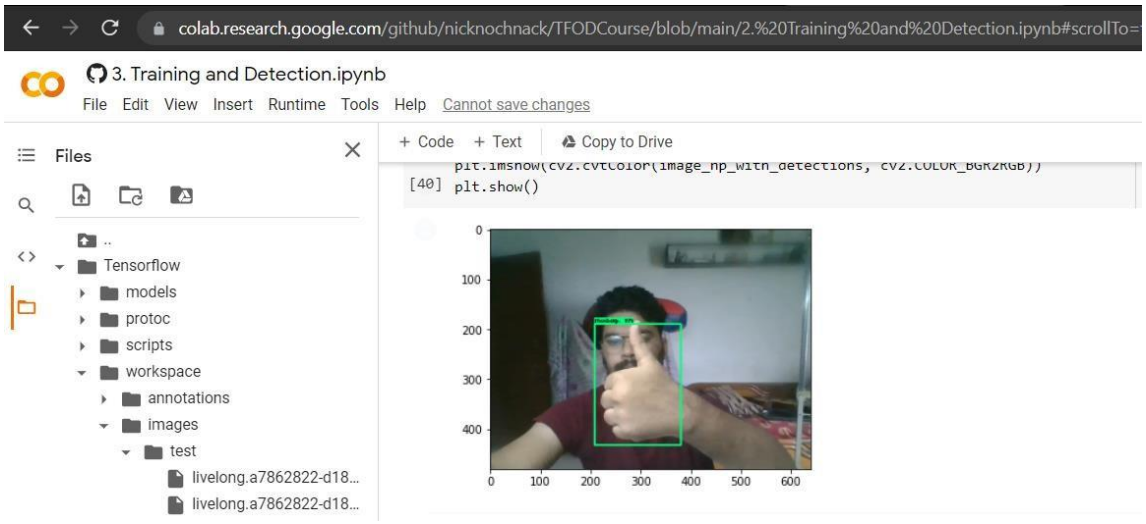


Image:



Tensorflow object detection :-

Screen shot: -

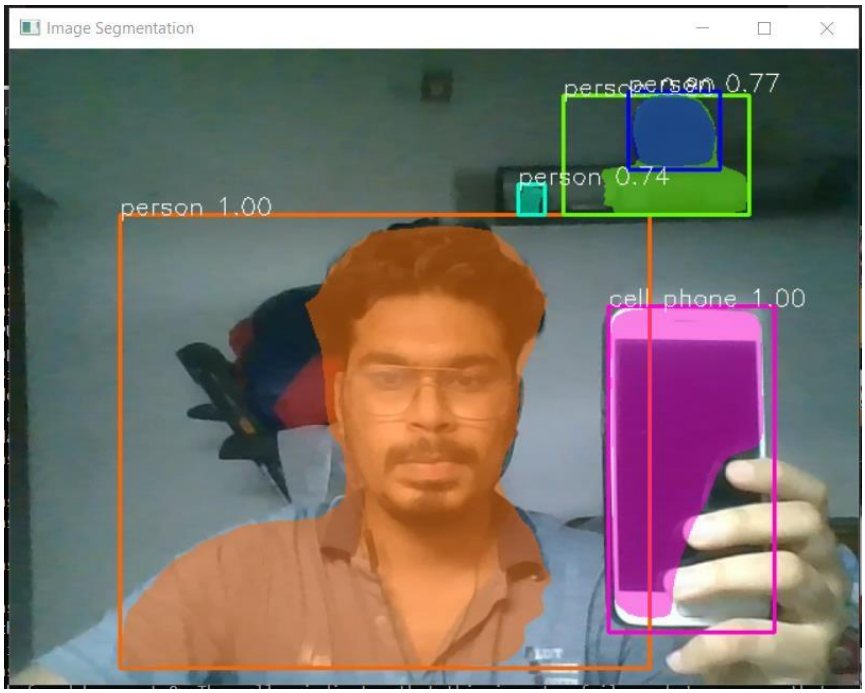


Performance: -

```
INFO:tensorflow:Step 1800 per-step time 0.092s
I0628 18:48:50.044215 139874177181568 model_lib_v2.py:700] Step 1800 per-step time 0.092s
INFO:tensorflow:{'Loss/classification_loss': 0.05712028,
 'Loss/localization_loss': 0.027453441,
 'Loss/regularization_loss': 0.14465624,
 'Loss/total_loss': 0.22922996,
 'learning_rate': 0.0799474}
I0628 18:48:50.044508 139874177181568 model_lib_v2.py:701] {'Loss/classification_loss': 0.05712028,
 'Loss/localization_loss': 0.027453441,
 'Loss/regularization_loss': 0.14465624,
 'Loss/total_loss': 0.22922996,
 'learning_rate': 0.0799474}
INFO:tensorflow:Step 1900 per-step time 0.093s
I0628 18:48:59.313402 139874177181568 model_lib_v2.py:700] Step 1900 per-step time 0.093s
INFO:tensorflow:{'Loss/classification_loss': 0.06344708,
 'Loss/localization_loss': 0.020403342,
 'Loss/regularization_loss': 0.14391804,
 'Loss/total_loss': 0.22776847,
 'learning_rate': 0.07993342}
I0628 18:48:59.313693 139874177181568 model_lib_v2.py:701] {'Loss/classification_loss': 0.06344708,
 'Loss/localization_loss': 0.020403342,
 'Loss/regularization_loss': 0.14391804,
 'Loss/total_loss': 0.22776847,
 'learning_rate': 0.07993342}
INFO:tensorflow:Step 2000 per-step time 0.094s
I0628 18:49:08.694043 139874177181568 model_lib_v2.py:700] Step 2000 per-step time 0.094s
INFO:tensorflow:{'Loss/classification_loss': 0.052716207,
 'Loss/localization_loss': 0.010015787,
 'Loss/regularization_loss': 0.14319916,
 'Loss/total_loss': 0.20593116,
 'learning_rate': 0.07991781}
I0628 18:49:08.694345 139874177181568 model_lib_v2.py:701] {'Loss/classification_loss': 0.052716207,
 'Loss/localization_loss': 0.010015787,
 'Loss/regularization_loss': 0.14319916,
 'Loss/total_loss': 0.20593116,
 'learning_rate': 0.07991781}
```

MASKED RCNN using Pixellib:-

Screen shot: -



Performance: -

6. Implementation:

System Configurations:

Processor: Intel(R) Core (TM) i7-7500U CPU @ 2.70GHz 2.90 GHz

Installed RAM: 16GB SSD4

Graphics: AMD Radeon 530 / Intel HD Graphics 620

Python Libraries Used:

- NumPy (version – 1.20.2)
- opencv-Python (version - 4.5.2.54) - virtual environment /opencv (version - 4.0.1) – conda environment

HOG Classifier for Pedestrian Detection:

Code:

```
import cv2
import imutils
import time

# Initializing the HOG person
# detector
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

cap = cv2.VideoCapture('vid.mp4')

# used to record the time when we processed last frame
prev_frame_time = 0

# used to record the time at which we processed current frame
new_frame_time = 0

while cap.isOpened():
    # Reading the video stream
    ret, image  =  cap.read()
    if ret:
        image = imutils.resize(image,
                               width=min(400, image.shape[1]))

        # Detecting all the regions
        # in the Image that has a
        # pedestrians inside it
        (regions, _) = hog.detectMultiScale(image,
```

```
                                            winStride=(4, 4),
                                            padding=(4, 4),
                                            scale=1.05)

        # Drawing the regions in the
        # Image
        for (x, y, w, h) in regions:
            cv2.rectangle(image, (x, y),
                          (x + w, y + h),
                          (255, 0, 0), 2)
            cv2.putText(image, 'Person', (x, y-
10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0,255,12), 2)

        # Showing the output Image
        cv2.imshow("Image", image)
        if cv2.waitKey(25) & 0xFF == ord('q'):
            break
    else:
        break

    new_frame_time = time.time()
    fps = 1/(new_frame_time-prev_frame_time)
    prev_frame_time = new_frame_time
    fps = int(fps)
    print(fps)

cap.release()
cv2.destroyAllWindows()
```

HAAR Cascade classifier for Face and eyes Detection:

Code:

```
import numpy as np
import cv2
import imutils
import time

cap = cv2.VideoCapture('video.mp4')

# used to record the time when we processed last frame
prev_frame_time = 0

# used to record the time at which we processed current frame
new_frame_time = 0

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcasca
de_frontalface_default.xml')
```

```
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascad
e_eye.xml')

while True:
    ret, frame = cap.read()
    if ret:
        frame = imutils.resize(frame,
                               width=min(800, frame.shape[1]))
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)

        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 5)
            cv2.putText(frame, 'Person', (x, y-
10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0,123,123), 2)
            roi_gray = gray[y:y+w, x:x+w]
            roi_color = frame[y:y+w, x:x+w]
            eyes = eye_cascade.detectMultiScale(roi_gray, 1.3, 5)
            for (ex, ey, ew, eh) in eyes:
                cv2.rectangle(roi_color, (ex, ey), (ex+ew, ey+eh), (0,
255, 125), 5)

        cv2.imshow('frame', frame)

        if cv2.waitKey(1) == ord('q'):
            break

    new_frame_time = time.time()
    fps = 1/(new_frame_time-prev_frame_time)
    prev_frame_time = new_frame_time
    fps = int(fps)
    print(fps)


cap.release()
cv2.destroyAllWindows()
```

System Configuration:

Processor: Intel(R) Core (TM) i5-8250U CPU @ 1.60GHz 1.80 GHz

Installed RAM: 8.00 GB

Graphics Card: AMD Radeon 520 / Intel UHD Graphics 620

Python Libraries Used:

- NumPy (version – 1.20.2)
- opencv-Python (version - 4.5.2.54) - virtual environment /opencv (version - 4.0.1) – conda environment

Python version: - 3.8.10

Additional requirements:

- coco.names - names of classes in coco dataset
- model configuration file: - yolov3.cfg [for yolo]
- model weights file: - yolov3.weights [for yolo]
- model configuration file: - ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt (Reference: - ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt · GitHub) [for ssd]
- model weights file: - frozen_inference_graph.pb (Reference: - Object Detection using OpenCV (Inference) | Kaggle) [for ssd]

You only Look Once v3:

Code:

```
import cv2
import numpy as np

cap=cv2.VideoCapture(0)
whT=250 #keep it between 220 and 320 for better frame rate
hgT=250
confidenceThreshold=0.5
nmsThreshold=0.3 #lower ->more aggressive and less no of boxes

classesFile='coco.names' #file containing the names of classes in coco dataset
classNames=[] #80 different names

with open(classesFile,'rt') as f:
    classNames=f.read().rstrip('\n').split('\n')
#print(classNames)
modelConfiguration='yolov3.cfg' #modelConfiguration='yolov3-tiny.cfg'
modelWeights='yolov3.weights'   #modelWeights='yolov3-tiny.weights'

net=cv2.dnn.readNetFromDarknet(modelConfiguration,modelWeights) #creating network
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)
net.setPreferableTarget(cv2.dnn.DNN_TARGET_CPU)


cap.set(3,640)
cap.set(4,480)
#cap.set(10,100) #brightness


def findObjects(outputs,img):
    ht,wt,ct=img.shape #height, width and channels of the image
    boundingBox=[] #it'll contain values of x,y,width and height
    classIds=[]
    confs=[]

    for output in outputs: # 3 outputs
        for det in output:
            scores=det[5:]
            classID=np.argmax(scores) #find index of max value
            confidence=scores[classID] #get the max value/probability
            if confidence>confidenceThreshold:
```

```
            if confidence>confidenceThreshold:
                w,h=int(det[2]*wt), int(det[3]*ht) #multipyling with width and height of image to get the pixel value
                x,y=int((det[0]*wt)-w/2),int((det[1]*ht)-h/2)
                boundingBox.append([x,y,w,h])
                classIds.append(classID)
                confs.append(float(confidence))
    print(len(boundingBox))
    indices=cv2.dnn.NMSBoxes(boundingBox,confs,confidenceThreshold,nmsThreshold) #non-max suppression function - elimnates overlapping boxes

    for i in indices:
        i=i[0]
        box=boundingBox[i]
        x,y,w,h=box[0],box[1],box[2],box[3]
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,255),2)
        cv2.putText(img,f'{classNames[classIds[i]].upper()} {int(confs[i]*100)}%',(x,y-10),cv2.FONT_HERSHEY_COMPLEX,0.6,(255,0,255),2)

while cap.isOpened():
    timer=cv2.getTickCount()
    success,img=cap.read()
    #path="w motors lykan hypersport.jpg"
    #img=cv2.resize(cv2.imread(path),(640,480))

    #path="1.jpg"
    #img=cv2.resize(cv2.imread(path),(1280,720))

    fps=cv2.getTickFrequency()/(cv2.getTickCount()-timer)
    cv2.putText(img,str(int(fps)),(75,50),cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,0,255),2)

    blob=cv2.dnn.blobFromImage(img,1/255,(whT,hgT),[0,0,0],1,crop=False) #converting image to blob (blob - type of format of image accepted by the network)
    net.setInput(blob)

    layerNames=net.getLayerNames()
    #print(layerNames)
    outputNames=[layerNames[i[0]-1] for i in net.getUnconnectedOutLayers()]
    #print(outputNames)
    #print(net.getUnconnectedOutLayers())

    outputs=net.forward(outputNames) #send the image as forward pass to the network
    #print(outputs[0].shape)
    #print(outputs[1].shape)
    #print(outputs[2].shape)
```

```
    fps=cv2.getTickFrequency()/(cv2.getTickCount()-timer)
    cv2.putText(img,str(int(fps)),(75,50),cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,0,255),2)

    blob=cv2.dnn.blobFromImage(img,1/255,(whT,hgT),[0,0,0],1,crop=False) #converting image to blob (blo
    net.setInput(blob)

    layerNames=net.getLayerNames()
    #print(layerNames)
    outputNames=[layerNames[i[0]-1] for i in net.getUnconnectedOutLayers()]
    #print(outputNames)
    #print(net.getUnconnectedOutLayers())

    outputs=net.forward(outputNames) #send the image as forward pass to the network
    #print(outputs[0].shape)
    #print(outputs[1].shape)
    #print(outputs[2].shape)
    #print(outputs[0][0])

    findObjects(outputs,img)

    cv2.imshow("Video",img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

SSD:

```
import cv2
import numpy as np

#path="w motors lykan hypersport.jpg"
#img=cv2.resize(cv2.imread(path),(640,480))

confThreshold=0.5 #threshold to detect object

cap=cv2.VideoCapture(0)
cap.set(3,640)
cap.set(4,480)

config_file='ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
weights_file='frozen_inference_graph.pb'

classesFile='coco.names'
classNames=[]
with open(classesFile,'rt') as f:
    classNames=f.read().rstrip('\n').split('\n')

#print(classNames)
#print(len(classNames))


#net=cv2.dnn.readNetFromTensorflow(weights_file,config_file)

net=cv2.dnn_DetectionModel(weights_file,config_file)
net.setInputSize(320,320)
net.setInputScale(1.0/127.5)
net.setInputMean((127.5,127.5,127.5))
net.setInputSwapRB(True)


while cap.isOpened():
    success,img=cap.read()
    #net.setInput(cv2.dnn.blobFromImage(img, size=(320, 320), swapRB=True, crop=False))
    #output=net.forward()

    classIds,confs,bbox=net.detect(img,confThreshold=0.5)
    #print(classIds)#bbox)
```

```
while cap.isOpened():
    success,img=cap.read()
    #net.setInput(cv2.dnn.blobFromImage(img, size=(320, 320), swapRB=True, crop=False))
    #output=net.forward()

    classIds,confs,bbox=net.detect(img,confThreshold=0.5)
    #print(classIds)#bbox)
    print(len(bbox))

    if len(classIds)!=0:
        for classId,confidence,box in zip(classIds.flatten(),confs.flatten(),bbox):
            cv2.rectangle(img,box,color=(0,255,0),thickness=2)
            cv2.putText(img,classNames[classId-1].upper(),(box[0]+10,box[1]+30),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)
            cv2.putText(img,str(round(confidence*100,2)),(box[0]+10,box[1]+200),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)

    cv2.imshow("Video",img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

MY System Configurations:

Processor: Intel(R) Core(TM) i5-8300h CPU @ 2.30GHz - 3.80 GHz
Installed RAM: 8GB SSD4
Graphics: Nvidia GEFORCE GTX 1050/ Intel HD Graphics 620
OS :- Microsoft Windows 10

Dependencies:-

Visual studio c++ build tools 2015
https://visualstudio.microsoft,com/vs/community/

CUDA and CUDNN (needs graphics card)

Packages required:-

absl-py==0.13.0
astunparse==1.6.3
backcall==0.2.0
cachetools==4.2.2
certifi==2021.5.30
chardet==4.0.0
colorama==0.4.4
cycler==0.10.0
Cython==0.29.23
decorator==5.0.9
flatbuffers==1.12
gast==0.4.0
gin==0.1.6
gin-config==0.1.1
google-auth==1.31.0
google-auth-oauthlib==0.4.4
google-pasta==0.2.0
grpcio==1.34.1
h5py==3.1.0
idna==2.10
ipykernel==5.5.5
ipython==7.24.1
ipython-genutils==0.2.0
jedi==0.18.0
jupyter-client==6.1.12
jupyter-core==4.7.1
keras-nightly==2.5.0.dev2021032900
Keras-Preprocessing==1.1.2
kiwisolver==1.3.1

lvis==0.5.3
lxml==4.6.3
Markdown==3.3.4
matplotlib==3.4.2
matplotlib-inline==0.1.2
numpy==1.19.5
oauthlib==3.1.1
object-detection==0.1
opencv-python==4.5.2.54
opt-einsum==3.3.0
pandas==1.3.0rc1
parso==0.8.2
pickleshare==0.7.5
Pillow==8.2.0
prompt-toolkit==3.0.19
protobuf==3.17.3
pyasn1==0.4.8
pyasn1-modules==0.2.8
pycocotools @
git+https://github.com/philferriere/cocoapi.git@2929bd2ef6b451054755dfd7ceb09278f935f7
ad#subdirectory=PythonAPI
Pygments==2.9.0
pyparsing==2.4.7
PyQt5==5.15.4
PyQt5-Qt5==5.15.2
PyQt5-sip==12.9.0
python-dateutil==2.8.1
pytz==2021.1
pywin32==301
PyYAML==5.4.1
pyzmq==22.1.0
requests==2.25.1
requests-oauthlib==1.3.0
rsa==4.7.2
scipy==1.7.0rc2
six==1.15.0
-e
git+https://github.com/tensorflow/models@5ad16f952885c86ca0aa31a8eb3737ab7bb23ee1#
egg=slim&subdirectory=research\slim
tensorboard==2.5.0
tensorboard-data-server==0.6.1
tensorboard-plugin-wit==1.8.0
tensorflow==2.5.0
tensorflow-addons==0.13.0
tensorflow-estimator==2.5.0
tensorflow-gpu==2.5.0
termcolor==1.1.0
tf-models-official==2.5.0
tf-slim==1.1.0
tornado==6.1
traitlets==5.0.5
typeguard==2.12.1
typing-extensions==3.7.4.3
urllib3==1.26.5
wcwidth==0.2.5
Werkzeug==2.0.1
wget==3.2
wrapt==1.12.1

TensorFlow Object Detection 2 With SSDNET

Steps :-

1) Step 1:-

Cloning the Github repository
https://github.com/nicknochnack/TFODCourse

2) Step 2:-
Creating and activating Virtual Environment

```
python -m venv tfod
.\tfod\Scripts\activate
```

3) Step 3:- Install dependencies and add virtual environment to the Python Kernel

```
python -m pip install --upgrade pip

pip install ipykernel

python -m ipykernel install --user --name=tfodj
```

4) Step 4:- Collecting Images from Webcam (importing dependencies)

Code

```
# Import opencv
import cv2

# Import uuid name our images uniquely
import uuid

# Import Operating System using for file  paths
import os

# Import time
import time
```

Screenshot:-

1. Import Dependencies

```
!pip install opencv-python
```

```
Requirement already satisfied: opencv-python in c:\users\deepn\anaconda3\lib\site-packages (4.5.2.54)
Requirement already satisfied: numpy>=1.17.3 in c:\users\deepn\anaconda3\lib\site-packages (from opencv-python) (1.20.1)
```

```
# Import opencv
import cv2

# Import uuid name our images uniquely
import uuid

# Import Operating System using for file  paths
import os

# Import time
import time
```

5)Step 5:- setting up folders where to store the webcam images

code

```
IMAGES_PATH = os.path.join('Tensorflow', 'workspace', 'images', 'collectedimages')
print(IMAGES_PATH)
Os.name
if not os.path.exists(IMAGES_PATH): #checks what type of operating syestem its using
   if os.name == 'posix':     #linux
      !mkdir -p {IMAGES_PATH}
   if os.name == 'nt':         #windows
       !mkdir {IMAGES_PATH}
for label in labels:
   path = os.path.join(IMAGES_PATH, label)
   if not os.path.exists(path):
      !mkdir {path}
```

Screenshot:-

3. Setup Folders ¶

```
In [4]: IMAGES_PATH = os.path.join('Tensorflow', 'workspace', 'images', 'collectedimages')

In [5]: print(IMAGES_PATH)

        Tensorflow\workspace\images\collectedimages

In [8]: os.name

Out[8]: 'nt'

In [6]: if not os.path.exists(IMAGES_PATH):   #checks what type of operating syestem its using
            if os.name == 'posix':     #linux
                !mkdir -p {IMAGES_PATH}
            if os.name == 'nt':          #windows
                 !mkdir {IMAGES_PATH}
        for label in labels:
            path = os.path.join(IMAGES_PATH, label)
            if not os.path.exists(path):
                !mkdir {path}
```

6)Step 6:- Capturing the Images from Webcam

Code:

```
for label in labels:       #this gonna loop through all the labels
  cap = cv2.VideoCapture(0)     #connect the webcam
  print('Collecting images for {}'.format(label)) #print out which label we are collecting
  time.sleep(5)  #time break for 5 secs
  for imgnum in range(number_imgs): #this will loop through the number of images
    print('Collecting image {}'.format(imgnum))  # collecting for specific label
    ret, frame = cap.read()  #specific frame
    imgname = os.path.join(IMAGES_PATH,label,label+'.'+'{}.jpg'.format(str(uuid.uuid1())))
#create a new image and plave inside a new path
    #also this will give us the extension with unique number
    cv2.imwrite(imgname, frame)
    cv2.imshow('frame', frame)
    time.sleep(2)

    if cv2.waitKey(1) & 0xFF == ord('q'):
      break
cap.release()
cv2.destroyAllWindows()
```

4. Capture Images

```
In [7]: for label in labels:        #this gonna loop through all the labels
            cap = cv2.VideoCapture(0)     #connect the webcam
            print('Collecting images for {}'.format(label))    #print out which label we are collecting
            time.sleep(5)    #time break for 5 secs
            for imgnum in range(number_imgs):    #this will loop through the number of images
                print('Collecting image {}'.format(imgnum))   # collecting for specific label
                ret, frame = cap.read()   #specific frame
                imgname = os.path.join(IMAGES_PATH,label,label+'.'+'{}.jpg'.format(str(uuid.uuid1())))   #create a new image and plave i
                #also this will give us the extension with unique number
                cv2.imwrite(imgname, frame)
                cv2.imshow('frame', frame)
                time.sleep(2)

                if cv2.waitKey(1) & 0xFF == ord('q'):
                    break
        cap.release()
        cv2.destroyAllWindows()
```

```
Collecting images for thumbsup
Collecting image 0
Collecting image 1
Collecting image 2
Collecting image 3
Collecting image 4
```

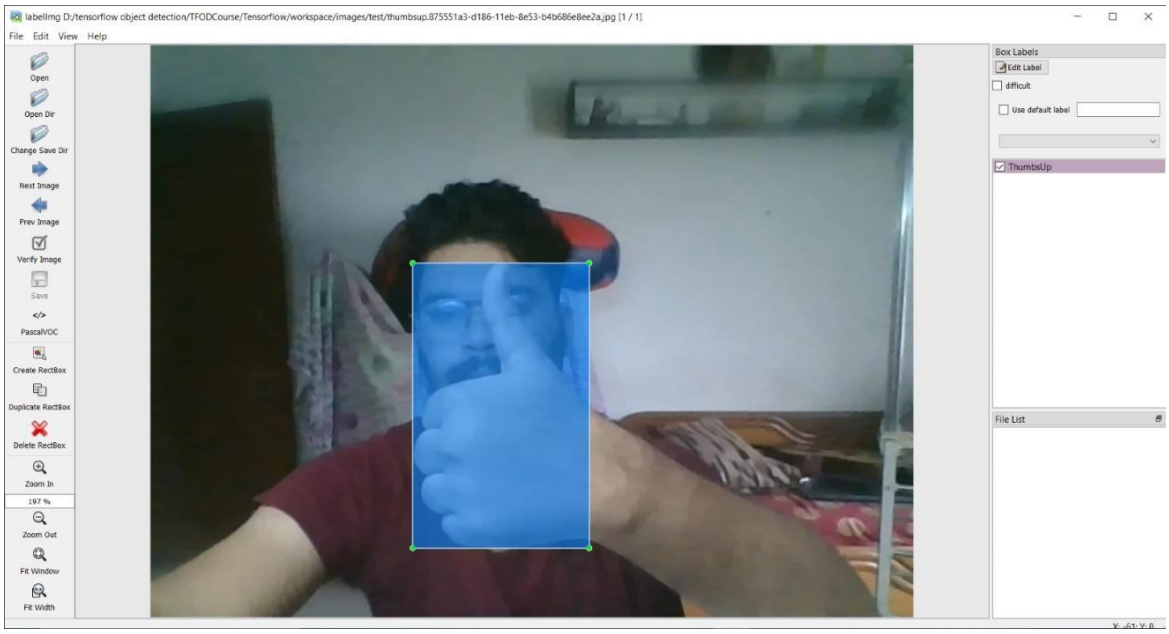
7) Step 7:- Labelling the image using LabelImg

Code

```
#installing
!pip install --upgrade pyqt5 lxml

#joining to the path
LABELIMG_PATH = os.path.join('Tensorflow', 'labelimg')

#if in case not present then clone
if not os.path.exists(LABELIMG_PATH):
  !mkdir {LABELIMG_PATH}
  !git clone https://github.com/tzutalin/labelImg {LABELIMG_PATH}

#compiling resourses for specific os
if os.name == 'posix':
  !cd {LABELIMG_PATH} && make qt5py3
if os.name =='nt':
  !cd {LABELIMG_PATH} && pyrcc5 -o libs/resources.py resources.qrc


#running the script and loading labelimg
!cd {LABELIMG_PATH} && python labelImg.py
```

This will run the label img script which will be like this

Screenshot:-




```
In [10]: LABELIMG_PATH = os.path.join('Tensorflow', 'labelimg')

In [11]: if not os.path.exists(LABELIMG_PATH):
             !mkdir {LABELIMG_PATH}
             !git clone https://github.com/tzutalin/labelImg {LABELIMG_PATH}

         Cloning into 'Tensorflow\labelimg'...

In [12]: if os.name == 'posix':
             !cd {LABELIMG_PATH} && make qt5py3
         if os.name =='nt':
             !cd {LABELIMG_PATH} && pyrcc5 -o libs/resources.py resources.qrc

In [13]: !cd {LABELIMG_PATH} && python labelImg.py

         Image:D:\tensorflow object detection\TFODCourse\Tensorflow\workspace\images\collectedimages\livelong\livelong.a2a55ca6-d186-11e
         b-a5ba-b4b686e8ee2a.jpg -> Annotation:D:/tensorflow object detection/TFODCourse/Tensorflow/workspace/images/collectedimages/liv
         elong/livelong.a2a55ca6-d186-11eb-a5ba-b4b686e8ee2a.xml
         Image:D:\tensorflow object detection\TFODCourse\Tensorflow\workspace\images\collectedimages\livelong\livelong.a3dd994b-d186-11e
         b-9141-b4b686e8ee2a.jpg -> Annotation:D:/tensorflow object detection/TFODCourse/Tensorflow/workspace/images/collectedimages/liv
         elong/livelong.a3dd994b-d186-11eb-9141-b4b686e8ee2a.xml
         Image:D:\tensorflow object detection\TFODCourse\Tensorflow\workspace\images\collectedimages\livelong\livelong.a64f700e-d186-11e
         b-bdae-b4b686e8ee2a.jpg -> Annotation:D:/tensorflow object detection/TFODCourse/Tensorflow/workspace/images/collectedimages/liv
```

After this making two folders as test and train and keep the ratio of images with the XML file generated using labelImg in ratio of 20:80 respectively.

8)Step 8 :- Setting up the path

Here the step is done in COLAB as some dependency conflicted with the jupyter system and also this was running under the GPU acceleration.
So cloning the same repository again as done in step 1.

Here the algo used for object detection is SSD_MOBILENET_V2_FPNLITE_320x320

Code :-

```
import os

#the name and alloting the url of the model we are going to use
```

```
CUSTOM_MODEL_NAME = 'my_ssd_mobnet'
PRETRAINED_MODEL_NAME = 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8'
PRETRAINED_MODEL_URL =
'http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz'
TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
LABEL_MAP_NAME = 'label_map.pbtxt'


#path to be configured in respective directory
paths = {
   'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),
   'SCRIPTS_PATH': os.path.join('Tensorflow','scripts'),
   'APIMODEL_PATH': os.path.join('Tensorflow',' we hamodels'),
   'ANNOTATION_PATH': os.path.join('Tensorflow', 'workspace','annotations'),
   'IMAGE_PATH': os.path.join('Tensorflow', 'workspace','images'),
   'MODEL_PATH': os.path.join('Tensorflow', 'workspace','models'),
   'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow', 'workspace','pre-trained-models'),
   'CHECKPOINT_PATH': os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME),
   'OUTPUT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME,
'export'),
   'TFJS_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME,
'tfjsexport'),
   'TFLITE_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME,
'tfliteexport'),
   'PROTOC_PATH':os.path.join('Tensorflow','protoc')
}


#files such as pipeline config ,tfrecord are alos placed to the path workspace
files = {
   'PIPELINE_CONFIG':os.path.join('Tensorflow', 'workspace','models', CUSTOM_MODEL_NAME,
'pipeline.config'),
   'TF_RECORD_SCRIPT': os.path.join(paths['SCRIPTS_PATH'], TF_RECORD_SCRIPT_NAME),
   'LABELMAP': os.path.join(paths['ANNOTATION_PATH'], LABEL_MAP_NAME)
}

#confirming whether path has been succesfully allocated or not
for path in paths.values():
   if not os.path.exists(path):
      if os.name == 'posix':
         !mkdir -p {path}
      if os.name == 'nt':
         !mkdir {path}
```

Screenshot:

```
import os
```

```
CUSTOM_MODEL_NAME = 'my_ssd_mobnet'
PRETRAINED_MODEL_NAME = 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8'
PRETRAINED_MODEL_URL = 'http://download.tensorflow.org/models/object_detection/t
TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
LABEL_MAP_NAME = 'label_map.pbtxt'
```

```
paths = {
    'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),
    'SCRIPTS_PATH': os.path.join('Tensorflow','scripts'),
    'APIMODEL_PATH': os.path.join('Tensorflow',' we hamodels'),
    'ANNOTATION_PATH': os.path.join('Tensorflow', 'workspace','annotations'),
    'IMAGE_PATH': os.path.join('Tensorflow', 'workspace','images'),
    'MODEL_PATH': os.path.join('Tensorflow', 'workspace','models'),
    'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow', 'workspace','pre-trained
    'CHECKPOINT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MO
    'OUTPUT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_
    'TFJS_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAM
    'TFLITE_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_N
    'PROTOC_PATH':os.path.join('Tensorflow','protoc')
 }
```

```
files = {
    'PIPELINE_CONFIG':os.path.join('Tensorflow', 'workspace','models', CUSTOM_MO
    'TF_RECORD_SCRIPT': os.path.join(paths['SCRIPTS_PATH'], TF_RECORD_SCRIPT_NAM
    'LABELMAP': os.path.join(paths['ANNOTATION_PATH'], LABEL_MAP_NAME)
}
```

```
for path in paths.values():
    if not os.path.exists(path):
        if os.name == 'posix':
            !mkdir -p {path}
        if os.name == 'nt':
            !mkdir {path}
```

9) Step 9:- Downloading Tensorflow Pretrained models from TensorFlow model zoo(Which consists TFOD(TensorFlow Object Detection))

Here to download the tensorflow model zoo we need to download the protoc files as they are actually helpfull in binding .

Code:-

```
if os.name=='nt':
  !pip install wget
  import wget


if not os.path.exists(os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection')):
  !git clone https://github.com/tensorflow/models {paths['APIMODEL_PATH']}


# Install Tensorflow Object Detection
if os.name=='posix':
  !apt-get install protobuf-compiler
```

```
!cd Tensorflow/models/research && protoc object_detection/protos/*.proto --python_out=. && cp object_detection/packages/tf2/setup.py . && python -m pip install .

if os.name=='nt':
    url="https://github.com/protocolbuffers/protobuf/releases/download/v3.15.6/protoc-3.15.6-win64.zip"
    wget.download(url)
    !move protoc-3.15.6-win64.zip {paths['PROTOC_PATH']}
    !cd {paths['PROTOC_PATH']} && tar -xf protoc-3.15.6-win64.zip
    os.environ['PATH'] += os.pathsep + os.path.abspath(os.path.join(paths['PROTOC_PATH'], 'bin'))
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --python_out=. && copy object_detection\\packages\\tf2\\setup.py setup.py && python setup.py build && python setup.py install
    !cd Tensorflow/models/research/slim && pip install -e .
```

Screenshot:-

1. Download TF Models Pretrained Models from Tensorflow Model Zoo and Install TFOD

```
[ ]  # https://www.tensorflow.org/install/source_windows

[8]  if os.name=='nt':
         !pip install wget
         import wget

[9]  if not os.path.exists(os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection')):
         !git clone https://github.com/tensorflow/models {paths['APIMODEL_PATH']}
```

```
Cloning into 'Tensorflow/models'...
remote: Enumerating objects: 58054, done.
remote: Counting objects: 100% (461/461), done.
remote: Compressing objects: 100% (196/196), done.
remote: Total 58054 (delta 288), reused 414 (delta 265), pack-reused 57593
Receiving objects: 100% (58054/58054), 573.20 MiB | 37.93 MiB/s, done.
Resolving deltas: 100% (40245/40245), done.
```

Installing the TFOD with protoc / protobuf

```
# Install Tensorflow Object Detection
if os.name=='posix':
    !apt-get install protobuf-compiler
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --python_out=. && cp object_detection/packages/tf2/setup.py . && python -m pip install

if os.name=='nt':
    url="https://github.com/protocolbuffers/protobuf/releases/download/v3.15.6/protoc-3.15.6-win64.zip"
    wget.download(url)
    !move protoc-3.15.6-win64.zip {paths['PROTOC_PATH']}
    !cd {paths['PROTOC_PATH']} && tar -xf protoc-3.15.6-win64.zip
    os.environ['PATH'] += os.pathsep + os.path.abspath(os.path.join(paths['PROTOC_PATH'], 'bin'))
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --python_out=. && copy object_detection\\packages\\tf2\\setup.py setup.py && python se
    !cd Tensorflow/models/research/slim && pip install -e .
```

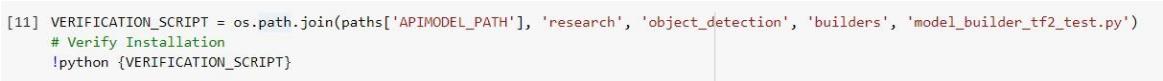
```
Requirement already satisfied: attrs>=18.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow-datasets->tf-models-official->object-detection==0.1) (21.2.0
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-packages (from kaggle>=1.3.9->tf-models-official->object-detection==0.1) (5.0.2)
Requirement already satisfied: google-api-core<2dev,>=1.21.0 in /usr/local/lib/python3.7/dist-packages (from google-api-python-client>=1.6.7->tf-models-official->ob
Requirement already satisfied: uritemplate<4dev,>=3.0.0 in /usr/local/lib/python3.7/dist-packages (from google-api-python-client>=1.6.7->tf-models-official->object-
Requirement already satisfied: google-auth-httplib2>=0.0.3 in /usr/local/lib/python3.7/dist-packages (from google-api-python-client>=1.6.7->tf-models-official->obje
Requirement already satisfied: google-auth>=1.16.0 in /usr/local/lib/python3.7/dist-packages (from google-api-python-client>=1.6.7->tf-models-official->object-detec
Collecting portalocker==2.0.0
  Downloading https://files.pythonhosted.org/packages/89/a6/3814b7107e0788040870e8825eebf214d72166adf656ba7d4bf14759a06a/portalocker-2.0.0-py2.py3-none-any.whl
Requirement already satisfied: cached-property; python_version < "3.8" in /usr/local/lib/python3.7/dist-packages (from h5py~=3.1.0->tensorflow>=2.5.0->tf-models-off
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.5->tensorflow>=2.5.0->tf-models-
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.5->tensorflow>=2.5.0->tf-models-official
```

Here a verification script is used to ensure that all necessary object detection files and packages are properly installed or not
If properly installed then it will return as ok

Code:-

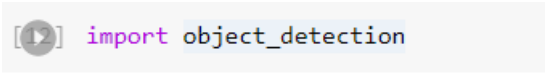
```
VERIFICATION_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection', 'builders', 'model_builder_tf2_test.py')
# Verify Installation
!python {VERIFICATION_SCRIPT}
```

Screenshot:-



10) Step 10
After installing the Tensorflow Object Detection alog with Protoc and verifying all files

Here the Object Detection api package can now be imported successfully .



11) Step 11: Now Creating Label map accordingly w.r.t labelImg which is used in xml scripting
Here the image file woth xml annotations are now label map and are given path

Code:-

```
labels = [{'name':'ThumbsUp', 'id':1}, {'name':'ThumbsDown', 'id':2}, {'name':'ThankYou', 'id':3}, {'name':'LiveLong', 'id':4}]

with open(files['LABELMAP'], 'w') as f:
  for label in labels:
    f.write('item { \n')
    f.write('\tname:\'{}\'\n'.format(label['name']))
    f.write('\tid:{}\n'.format(label['id']))
    f.write('}\n')
```

Screenshot:-

12) Step 12 :- Moving the tensorflow respective model into main dir

Code:-

```
#to move
if os.name == 'nt':
   wget.download(PRETRAINED_MODEL_URL)
   !move {PRETRAINED_MODEL_NAME+'.tar.gz'} {paths['PRETRAINED_MODEL_PATH']}
   !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf {PRETRAINED_MODEL_NAME+'.tar.gz'}
```

Screenshot :-

```
[13] if os.name =='posix':
         !wget {PRETRAINED_MODEL_URL}
         !mv {PRETRAINED_MODEL_NAME+'.tar.gz'} {paths['PRETRAINED_MODEL_PATH']}
         !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf {PRETRAINED_MODEL_NAME+'.tar.gz'}
     if os.name == 'nt':
         wget.download(PRETRAINED_MODEL_URL)
         !move {PRETRAINED_MODEL_NAME+'.tar.gz'} {paths['PRETRAINED_MODEL_PATH']}
         !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf {PRETRAINED_MODEL_NAME+'.tar.gz'}

     --2021-06-28 18:43:02--  http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz
     Resolving download.tensorflow.org (download.tensorflow.org)... 142.250.81.208, 2607:f8b0:4004:82f::2010
     Connecting to download.tensorflow.org (download.tensorflow.org)|142.250.81.208|:80... connected.
     HTTP request sent, awaiting response... 200 OK
     Length: 20515344 (20M) [application/x-tar]
     Saving to: 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz'

     ssd_mobilenet_v2_fp 100%[===================>]  19.56M  --.-KB/s    in 0.1s

     2021-06-28 18:43:02 (199 MB/s) - 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz' saved [20515344/20515344]

     ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/
     ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint/
     ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint/ckpt-0.data-00000-of-00001
     ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint/checkpoint
     ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint/ckpt-0.index
     ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/pipeline.config
     ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/saved_model/
     ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/saved_model/saved_model.pb
     ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/saved_model/variables/
     ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/saved_model/variables/variables.data-00000-of-00001
     ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/saved_model/variables/variables.index
```

13) Step 13 :- downloading or Cloning generating tensorflow record file such that this file will convert our xml annotations into file format.

```
#tf record tat converts your annotations into file formatif not
os.path.exists(files['TF_RECORD_SCRIPT']): !git clone
https://github.com/nicknochnack/GenerateTFRecord {paths['SCRIPTS_PATH']}

!python {files['TF_RECORD_SCRIPT']} -x {os.path.join(paths['IMAGE_PATH'], 'train')} -l
{files['LABELMAP']} -o {os.path.join(paths['ANNOTATION_PATH'], 'train.record')}
!python {files['TF_RECORD_SCRIPT']} -x {os.path.join(paths['IMAGE_PATH'], 'test')} -l
{files['LABELMAP']} -o {os.path.join(paths['ANNOTATION_PATH'], 'test.record')}
```

```
[16] if not os.path.exists(files['TF_RECORD_SCRIPT']):
         !git clone https://github.com/nicknochnack/GenerateTFRecord {paths['SCRIPTS_PATH']}

     Cloning into 'Tensorflow/scripts'...
     remote: Enumerating objects: 3, done.
     remote: Counting objects: 100% (3/3), done.
     remote: Compressing objects: 100% (2/2), done.
     remote: Total 3 (delta 0), reused 1 (delta 0), pack-reused 0
     Unpacking objects: 100% (3/3), done.

[17] !python {files['TF_RECORD_SCRIPT']} -x {os.path.join(paths['IMAGE_PATH'], 'train')} -l {files['LABELMAP']} -o {os.path.join(paths['ANNOTATION_PATH'], 'train.record')}
     !python {files['TF_RECORD_SCRIPT']} -x {os.path.join(paths['IMAGE_PATH'], 'test')} -l {files['LABELMAP']} -o {os.path.join(paths['ANNOTATION_PATH'], 'test.record')}

     Successfully created the TFRecord file: Tensorflow/workspace/annotations/train.record
     Successfully created the TFRecord file: Tensorflow/workspace/annotations/test.record
```

14) Step 14: -
The pipeline.config file will be copied to the training folder

Code:

```
if os.name == 'nt':
   !copy {os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME,
'pipeline.config')} {os.path.join(paths['CHECKPOINT_PATH'])}


import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format


config = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])

Config
```

Screenshot:-

4. Copy Model Config to Training Folder

```
[18] if os.name =='posix':
         !cp {os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'pipeline.config')} {os.path.join(paths['CHECKPOINT_PATH'])}
     if os.name == 'nt':
         !copy {os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'pipeline.config')} {os.path.join(paths['CHECKPOINT_PATH'])}
```

5. Update Config For Transfer Learning

```
[19] import tensorflow as tf
     from object_detection.utils import config_util
     from object_detection.protos import pipeline_pb2
     from google.protobuf import text_format
```

```
[20] config = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
```

```
config
```

Giving parameters to pipeling config file

```
pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "r") as f:
    proto_str = f.read()
    text_format.Merge(proto_str, pipeline_config)
```

```
] pipeline_config.model.ssd.num_classes = len(labels)
pipeline_config.train_config.batch_size = 4
pipeline_config.train_config.fine_tune_checkpoint = os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'checkpoint', 'ckpt-0')
pipeline_config.train_config.fine_tune_checkpoint_type = "detection"
pipeline_config.train_input_reader.label_map_path= files['LABELMAP']
pipeline_config.train_input_reader.tf_record_input_reader.input_path[:] = [os.path.join(paths['ANNOTATION_PATH'], 'train.record')]
pipeline_config.eval_input_reader[0].label_map_path = files['LABELMAP']
pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[:] = [os.path.join(paths['ANNOTATION_PATH'], 'test.record')]
```

```
] config_text = text_format.MessageToString(pipeline_config)
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "wb") as f:
    f.write(config_text)
```

15) Step 15:- Training the model

Now the main step of training the model with around 2000 steps

Code:

```
TRAINING_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection',
'model_main_tf2.py')
```

```
command = "python {} --model_dir={} --pipeline_config_path={} --
num_train_steps=2000".format(TRAINING_SCRIPT,
paths['CHECKPOINT_PATH'],files['PIPELINE_CONFIG'])
```

Screenshot:-

16) Step 16:-

Now after training our model has been trained accordingly and can be loaded with the checkpoint thanks to pipeline config

```
import os
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
from object_detection.utils import config_util


# Load pipeline config and build a detection model
configs = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
detection_model = model_builder.build(model_config=configs['model'], is_training=False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-3')).expect_partial()

@tf.function
def detect_fn(image):
    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
    detections = detection_model.postprocess(prediction_dict, shapes)
    return detections
```

```
[31] import os
     import tensorflow as tf
     from object_detection.utils import label_map_util
     from object_detection.utils import visualization_utils as viz_utils
     from object_detection.builders import model_builder
     from object_detection.utils import config_util
```

```
[33] # Load pipeline config and build a detection model
     configs = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
     detection_model = model_builder.build(model_config=configs['model'], is_training=False)

     # Restore checkpoint
     ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
     ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-3')).expect_partial()

     @tf.function
     def detect_fn(image):
         image, shapes = detection_model.preprocess(image)
         prediction_dict = detection_model.predict(image, shapes)
         detections = detection_model.postprocess(prediction_dict, shapes)
         return detections
```

17) Step17:-now detecting from an image from open cv here the code is given as below

Here we can use the open cv library through which the testing or the detecting phase can be successfully be implemented.

Code:

```
#importing necessary libraries
import cv2
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline


#assigning labelmap
category_index = label_map_util.create_category_index_from_labelmap(files['LABELMAP'])


#selecting image from test folder which we selected
IMAGE_PATH = os.path.join(paths['IMAGE_PATH'], 'test', 'livelong.a7862822-d186-11eb-a8c4-b4b686e8ee2a')




#the script which will give us the output
img = cv2.imread(IMAGE_PATH)
image_np = np.array(img)

input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
detections = detect_fn(input_tensor)

num_detections = int(detections.pop('num_detections'))
detections = {key: value[0, :num_detections].numpy()
          for key, value in detections.items()}
detections['num_detections'] = num_detections
```

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

label_id_offset = 1
image_np_with_detections = image_np.copy()

viz_utils.visualize_boxes_and_labels_on_image_array(
      image_np_with_detections,
      detections['detection_boxes'],
      detections['detection_classes']+label_id_offset,
      detections['detection_scores'],
      category_index,
      use_normalized_coordinates=True,
      max_boxes_to_draw=5,
      min_score_thresh=.8,
      agnostic_mode=False)

plt.imshow(cv2.cvtColor(image_np_with_detections, cv2.COLOR_BGR2RGB))
plt.show()
```

Screenshot:-

▾ 9. Detect from an Image

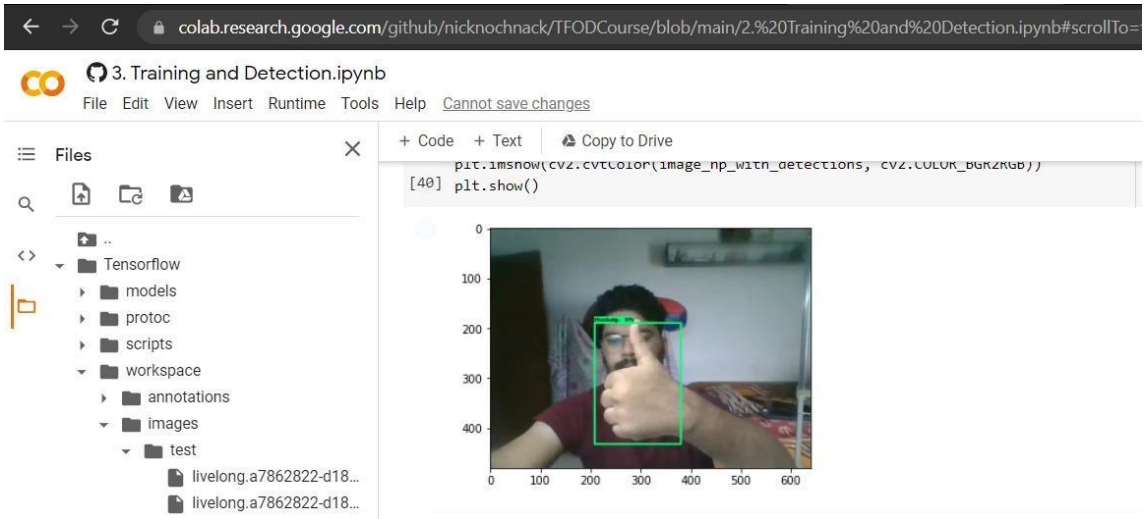
```
[37] import cv2
     import numpy as np
     from matplotlib import pyplot as plt
     %matplotlib inline
```

```
[38] category_index = label_map_util.create_category_index_from_labelmap(files['LABELMAP'])
```

```
[39] IMAGE_PATH = os.path.join(paths['IMAGE_PATH'], 'test', 'thumbsup.875551a3-d186-11eb-8e53-b4b686e8ee2a.jpg')
```

```
[40] img = cv2.imread(IMAGE_PATH)
     image_np = np.array(img)

     input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
     detections = detect_fn(input_tensor)

     num_detections = int(detections.pop('num_detections'))
     detections = {key: value[0, :num_detections].numpy()
                   for key, value in detections.items()}
     detections['num_detections'] = num_detections

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

     label_id_offset = 1
     image_np_with_detections = image_np.copy()

     viz_utils.visualize_boxes_and_labels_on_image_array(
                 image_np_with_detections,
                 detections['detection_boxes'],
                 detections['detection_classes']+label_id_offset,
                 detections['detection_scores'],
                 category_index,
                 use_normalized_coordinates=True,
                 max_boxes_to_draw=5,
                 min_score_thresh=.8,
                 agnostic_mode=False)

     plt.imshow(cv2.cvtColor(image_np_with_detections, cv2.COLOR_BGR2RGB))
     plt.show()
```

As one can see the bounding box is been formed properly



2)Object detection using PIXELLIB (on mask rcnn model)

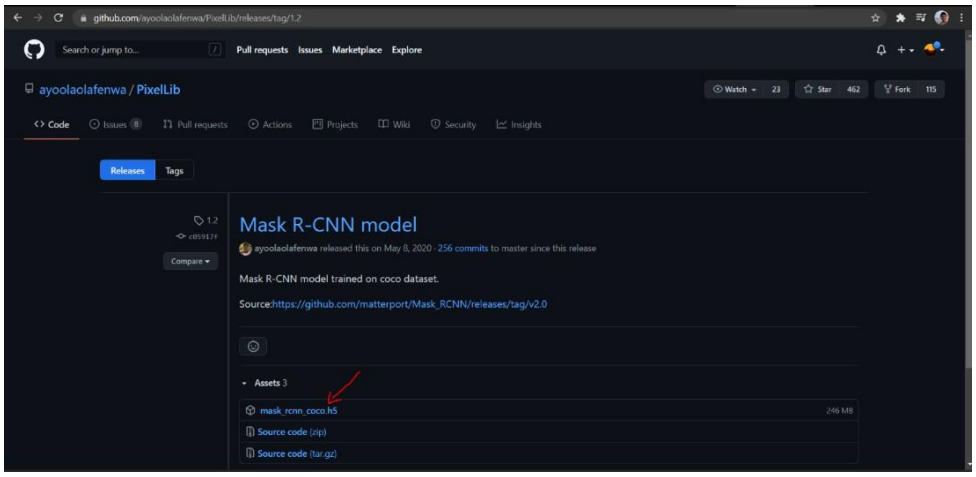
Packages required(to be installed in Virtual env):-

opencv-python
scikit-image
pillow
pixellib
tensorflow-gpu

Steps:-

1)step 1

Download the mask_rcnn_coo.h5 file from the site
https://github.com/ayoolaolafenwa/PixelLib/releases/download/1.2/mask_rcnn_coco.h5



2)step 2

Now writing a python code to run opencv programwhich executes this pretrained model

```
import numpy as np
import time
import cv2
import pixellib
from pixellib.instance import instance_segmentation
segment_image = instance_segmentation()
segment_image.load_model("mask_rcnn_coco.h5")
camera = cv2.VideoCapture(0)
```

```
# used to record the time when we processed last frame
prev_frame_time = 0

# used to record the time at which we processed current frame
new_frame_time = 0

while camera.isOpened():
   res,frame=camera.read()
   ### Apply Segmentation
   result=segment_image.segmentFrame(frame,show_bboxes=True)
   image=result[1]
   cv2.imshow('Image Segmentation',image)


   # Our operations on the frame come here
   gray = frame

   # resizing the frame size according to our need
   gray = cv2.resize(gray, (500, 300))



   # font which we will be using to display FPS
   font = cv2.FONT_HERSHEY_SIMPLEX
   # time when we finish processing for this frame
   new_frame_time = time.time()

   # Calculating the fps

   # fps will be number of frame processed in given time frame
   # since their will be most of time error of 0.001 second
   # we will be subtracting it to get more accurate result
   fps = 1/(new_frame_time-prev_frame_time)
   prev_frame_time = new_frame_time

   # converting the fps into integer
   fps = int(fps)

   # converting the fps to string so that we can display it on frame
   # by using putText function
   fps = str(fps)

   # putting the FPS count on the frame
   cv2.putText(gray, fps, (7, 70), font, 3, (100, 255, 0), 3, cv2.LINE_AA)

   # displaying the frame with fps
   cv2.imshow('frame', gray)

   if cv2.waitKey(5) & 0xFF==ord('q'):
      break



camera.release()
cv2.destroyAllWindows()
```

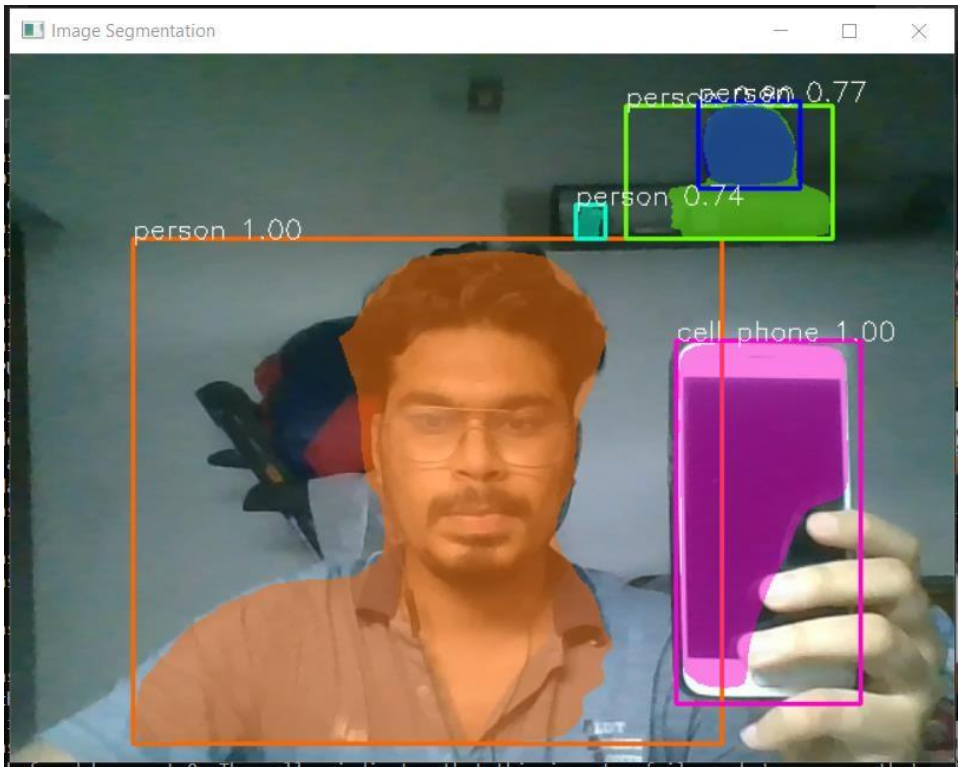
app.py - D:\pixellib\Image-Segmentation-Using-Pixellib-main\Live Video\app.py (3.9.5)

File Edit Format Run Options Window Help

```
import cv2
import pixellib
from pixellib.instance import instance_segmentation
segment_image = instance_segmentation()
segment_image.load_model("mask_rcnn_coco.h5")
camera = cv2.VideoCapture(0)

while camera.isOpened():
    res,frame=camera.read()
    ### Apply Segmentation
    result=segment_image.segmentFrame(frame,show_bboxes=True)
    image=result[1]
    cv2.imshow('Image Segmentation',image)

    if cv2.waitKey(5) & 0xFF==ord('q'):
        break

camera.release()
cv2.destroyAllWindows()
```

3)Step3:-

Now running the command in an virtual environment in CMD



7. Testing/ Evaluation

HAAR Cascade Classifier:

Accuracy: Detects frontal face and eyes accurately on,

- Selfie camera,
- Recorded video,
- Image (jpg/png),

Framerate:

- Maximum: 9
- Minimum: drops to 4 sometimes
- Average: 6

HOG Classifier:

Accuracy: Detects pedestrians on,

- Recorded video,
- Image (jpg/png),

As shown in screenshots, Accuracy is around 25-30% for crowded places.

100% for videos/ images with 4-5 people standing at a fair distance.

Framerate:

- Maximum: 6
- Minimum: drops to 2 sometimes
- Average: 4

For YOLO and SSD:

Algorithm evaluation:

Hyperparameter	SSD	YOLOv3
Training epochs	176,000	5000
Batch size	Between 6 and 10 images	Between 4 and 8 images
Optimizer	RMSProp	SGD Burn-In (in DarkNet [])
Learning rate	[0.001, 0.004]	[0.0001, 0.01]
Momentum	0.9	[0.8, 0.9]
Decay	0.9	[0.0003, 0.0005]

[Image 7.1 Main training parameters used for SSD and YOLOv3 network]

Sl.no.	Parameters	YOLO	SSD
1	Model name	You Only Look Once	Single Shot Multi-Box Detector
2	Speed	Low	High
3	Accuracy	80.3% High	72.1% Low
4	Time	0.84~0.9 sec/frame	0.17~0.23 sec/frame
5	Frame per second	45	59
6	Mean Average precision	0.358	0.251

[Image 7.2 Difference between YOLO and SSD]

- The figure above shows comparison between YOLO and SSD as regards to speed, accuracy, time, frame per second (FPS), Mean Average Precision (mAP), and whether they can be used for real time applications or not.

- It clearly shows that YOLO is better than the low accuracy and higher FPS SSD algorithm. At 416 X 416 YOLOv3 runs in 29 ms at 31.0 mAP almost as accurate as SSD but approximately 2.2 times faster that SSD. It can be seen clearly that a precise compromise was made to achieve this speed. Even after a low mAP, YOLO has an appropriate mAP to be used in real time applications, and it becomes obvious that it is the best algorithms in its class when used alongside high FPS accurate information.

TFOD SSD MOBILENET V2:-

- As shown below in screenshot per hundred steps it takes 0.092s on average for a step to get trained
- and as we increase the training rate the less the total loss appears

```
INFO:tensorflow:Step 1800 per-step time 0.092s
I0628 18:48:50.044215 139874177181568 model_lib_v2.py:700] Step 1800 per-step time 0.092s
INFO:tensorflow:{'Loss/classification_loss': 0.05712028,
 'Loss/localization_loss': 0.027453441,
 'Loss/regularization_loss': 0.14465624,
 'Loss/total_loss': 0.22922996,
 'learning_rate': 0.0799474}
I0628 18:48:50.044508 139874177181568 model_lib_v2.py:701] {'Loss/classification_loss': 0.05712028,
 'Loss/localization_loss': 0.027453441,
 'Loss/regularization_loss': 0.14465624,
 'Loss/total_loss': 0.22922996,
 'learning_rate': 0.0799474}
INFO:tensorflow:Step 1900 per-step time 0.093s
I0628 18:48:59.313402 139874177181568 model_lib_v2.py:700] Step 1900 per-step time 0.093s
INFO:tensorflow:{'Loss/classification_loss': 0.06344708,
 'Loss/localization_loss': 0.020403342,
 'Loss/regularization_loss': 0.14391804,
 'Loss/total_loss': 0.22776847,
 'learning_rate': 0.07993342}
I0628 18:48:59.313693 139874177181568 model_lib_v2.py:701] {'Loss/classification_loss': 0.06344708,
 'Loss/localization_loss': 0.020403342,
 'Loss/regularization_loss': 0.14391804,
 'Loss/total_loss': 0.22776847,
 'learning_rate': 0.07993342}
INFO:tensorflow:Step 2000 per-step time 0.094s
I0628 18:49:08.694043 139874177181568 model_lib_v2.py:700] Step 2000 per-step time 0.094s
INFO:tensorflow:{'Loss/classification_loss': 0.052716207,
 'Loss/localization_loss': 0.010015787,
 'Loss/regularization_loss': 0.14319916,
 'Loss/total_loss': 0.20593116,
 'learning_rate': 0.07991781}
I0628 18:49:08.694345 139874177181568 model_lib_v2.py:701] {'Loss/classification_loss': 0.052716207,
 'Loss/localization_loss': 0.010015787,
 'Loss/regularization_loss': 0.14319916,
 'Loss/total_loss': 0.20593116,
 'learning_rate': 0.07991781}
```

- As the real time video isnt working properly we learned from the tensorflow zoo that

SSD MobileNet V2 FPNLite 320x320	22	22.2	Boxes

- The second column describes the speed in ms the less the speed ,the better the working of the model and accuracy

```
Accumulating evaluation results...
DONE (t=0.02s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.725
 Average Precision  (AP) @[ IoU=0.50       | area=   all | maxDets=100 ] = 1.000
 Average Precision  (AP) @[ IoU=0.75       | area=   all | maxDets=100 ] = 1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.725
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.725
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.725
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.725
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.725
INFO:tensorflow:Eval metrics at step 2000
I0704 10:51:10.765097 139774760949632 model_lib_v2.py:1007] Eval metrics at step 2000
INFO:tensorflow:        + DetectionBoxes_Precision/mAP: 0.725000
I0704 10:51:10.766759 139774760949632 model_lib_v2.py:1010]     + DetectionBoxes_Precision/mAP: 0.725000
INFO:tensorflow:        + DetectionBoxes_Precision/mAP@.50IOU: 1.000000
I0704 10:51:10.767931 139774760949632 model_lib_v2.py:1010]     + DetectionBoxes_Precision/mAP@.50IOU: 1.000000
INFO:tensorflow:        + DetectionBoxes_Precision/mAP@.75IOU: 1.000000
```

- Here also the average precision and recall all around 2000 steps is aprox =0.725
- Thus SSD net with Tensorflow object detection is acting good in terms of performance and precision.

- Average frame rate according to source (with GPU) = 19.5fps
- https://www.dlology.com/blog/how-to-run-ssd-mobilenet-v2-object-detection-on-jetson-nano-at-20-fps/

MASKED RCNN PIXELLIB:-

- Here the model s used is masked rcnn which is comparable to faster rcnn

Method	mAP
Faster R-CNN-VGG	0.6958
Faster R-CNN-ResNet101	0.8976
Mask R-CNN-ResNet101	0.8996
Proposed	0.9063

- And here the Mask Rcnn is good in terms of edge detection
- But the Frame rate is way too low giving about average 1 to 3 fps as most of times it does semantic segmentation which indeed gives less fps .

- It is great in terms of detection but its slow in terms of FPS.

Mask R-CNN Inception ResNet V2 1024x1024	301	39.0/34.6	Boxes/Masks

- As we can see from this image the second column is speed in ms
- And its significantly higher than all of the object detection models .

8. Limitations and Future Enhancements:

HOG:

- HOG has a disadvantage that is very sensitive to image rotation. Therefore, HOG is not good choice for classification of textures or objects which can often be detected as rotated image.

Haar Cascade:

- The detector tends to be the most effective for frontal images of the face.
- Haar cascades are notoriously prone to false-positives — the Viola-Jones algorithm can easily report a face in an image when no face is present.

SSD:

- SSD performs worse than Faster R-CNN for small-scale objects. In SSD, small objects can only be detected in higher resolution layers (leftmost layers). But those layers contain low-level features, like edges or colour patches, that are less informative for classification.

YOLO:

- Comparatively low recall and more localization error compared to Faster R_CNN.
- Struggles to detect close objects because each grid can propose only 2 bounding boxes.
- Struggles to detect small objects.

Masked RCNN:

- Good interms of Detection but bad in terms of Framerate.
- Slow compared to other algorithms.
- Frame rate could be improved with better algorithms .

Future Enhancement

The Object detection can be run on GPU instead of CPU.

However, the constraint is that for a beginner setting up a deep neural network on a GPU can be a really harsh process.

Also, it doesn't work with all the GPUs but only with NVIDIA GPUs which are compatible with CUDA.

Also, some of this algorithms may be lighter in future so that the performance will not decrease significantly.

9. Conclusion:

1. HAAR Cascade:

   - Suggested to use for webcam/ selfie type facial detection/ recognition.
   - Can work in systems with less graphics processing advancements with decent framerate and accuracy.

2. HOG Classifier:

   - Suggested to use for static images (jpg/ png) or non-crowded videos for pedestrian detection.

3. Mask RCNN:

   - Can produce excellent results for all webcam, image or video file based on the neural network or dataset chosen.
   - Suggested to use on Machines with decent GPUs or CPUs for good framerate.

4. SSD and YOLO:

   - Produces best results.
   - High end GPUs recommended.

10. References:

Article and images for object detection and its basic algorithms:

- Object Detection : Simplified. Take a peek into the world of one of… | by Prakhar Ganesh | Towards Data Science
- Top 8 Algorithms For Object Detection One Must Know (analyticsindiamag.com)
- Introduction to Object Detection Algorithms (analyticsvidhya.com)
- Comparison of YOLOv3 and SSD Algorithms (ijert.org)

SSD vs. YOLO for Detection of Outdoor Urban Advertising Panels under Multiple Variabilities (nih.gov)

YouTube tutorial for HAAR Cascade classifier:

- https://www.youtube.com/watch?v=mPCZLOVTEc4&list=PLzMcBGfZo4-lUA8uGjeXhBUUzPYc6vZRn&index=8

On HOG Classifier:

- Research Paper by Dalal and Triggs

Article and Images for RCNN, Fast RCNN, Faster RCNN, Mask RCNN:

- https://medium.com/@umerfarooq_26378/from-r-cnn-to-mask-r-cnn-d6367b196cfd

Article and Images for SSD:

- https://developers.arcgis.com/python/guide/how-ssd-works/
- https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006

YOLO Object Detection pretrained model:

- https://www.kaggle.com/valentynsichkar/yolo-coco-data?select=yolov3.weights

SSD Object Detection pretrained model:

- ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt · GitHub
- Object Detection using OpenCV (Inference) | Kaggle

YouTube tutorial for YOLO object detection:

- YOLO v3 EASY METHOD | OpenCV Python (2020) - YouTube
- YOLO v3 EASY METHOD | OpenCV Python (2020) p.2 - YouTube
- YOLO v3 EASY METHOD | OpenCV Python (2020) p.3 - YouTube
- YOLO v3 EASY METHOD | OpenCV Python (2020) p.4 - YouTube

YouTube tutorial for SSD object detection:

- Object Detection OpenCV Python | Easy and Fast (2020) - YouTube

Limitations of object detection algorithms:

- YOLO : You Only Look Once - Real Time Object Detection - GeeksforGeeks
- SSD object detection: Single Shot MultiBox Detector for real-time processing | by Jonathan Hui | Medium
- OpenCV Haar Cascades - PyImageSearch

Future Enhancement:

- Real-Time Object detection on a CPU using one of the smartest Convolutional Neural Networks — The YOLO | by Emmytheo 24/7 | Medium

Tensorflow object detection:

Youtube references

- https://www.youtube.com/watch?v=yqkISICHH-U&t=12232s
- https://www.youtube.com/playlist?list=PLZoTAELRMXVNvTfHyJxPRcQkpV8ubBwHo

Github references

- https://github.com/nicknochnack/TFODCourse
- http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz
- https://github.com/protocolbuffers/protobuf/releases/download/v3.15.6/protoc-3.15.6-win64.zip
- https://github.com/tensorflow/models

Article ,references,necessities

- https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md
- https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/
- https://developer.nvidia.com/cuda-10.1-download-archive-base?target_os=Windows&target_arch=x86_64
- https://developer.nvidia.com/rdp/cudnn-archive

Masked RCNN using Pixellib:

Youtube references

- https://www.youtube.com/watch?v=xL_0sscWllM&list=PLZoTAELRMXVNvTfHyJxPRcQkpV8ubBwHo&index=9&t=201s

Github references

- https://github.com/ayoolaolafenwa/PixelLib
- https://github.com/ayoolaolafenwa/PixelLib/releases/download/1.2/mask_rcnn_coco.h5

Articles ,references,necessities

- https://www.kaggle.com/shawon10/object-detection-and-image-segmentation-pixellib
- https://towardsdatascience.com/image-segmentation-with-six-lines-0f-code-acb870a462e8

Bhaskaracharya National Institute for Space Applications and Geo-informatics
MeitY, Government of India
Phone: 079 - 23213081 Fax: 079 - 23213091
E-mail: info@bisag.gujarat.gov.in, website: https://bisag-n.in/

ISO 9001:2008
ISO 27001:2013
CMMI LEVEL-5

Report Verification Procedure

Date: 09/07/2021

Project Name: Comparison of Object Detection Models

Student Name & ID: 1. Tanmay Vaidya (H4)
2. Deepnil Vasava (H4)
3. Deep Patel (H4)

	Soft Copy	Hard Copy
Report Format:	☑	☑
Project Index:	☑	☑

Sign by Training Coordinator

Sign by Project Guide

89