**OBJECT DETECTION**

A Project Report submitted in partial fulfilment of the requirements for the award of the degree of

**Bachelor of Technology**

**In**

***Computer Science and Engineering***

Submitted by-

**NISHITA MAHESHWARI**

171500207

Under the guidance of

**Mr Vaibhav Diwan Sir**

Department of Computer Engineering & Application

**Institute of Engineering & Technology**



**GLA UNIVERSITY**

**MATHURA-281406, INDIA**

**Declaration**

I hereby declare that the work which is being presented in the B.tech. Project "**Object Detection using image**", in partial fulfilment of the requirements for the award of the Bachelor of Technology in Computer Science and Engineering and submitted to the Department of Computer Engineering and Applications of GLA University. Mathura is an authentic record of my own work carried under the supervision of **Mr Vaibhav Diwan Sir**, Asstt Professor. The contents of this project report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree

Signature of Candidate:

Name of Candidate: Nishita Maheshwari

Roll. No. : 171500207

Course: CSE

Year: III

Semester-V

Dept. of CEA, GLAU, Mathura i

**ACKNOWLEDGEMENT**

The success and partial outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I respect and thank Mr Vaibhav Diwan, for providing me an opportunity to do the project work in GLA University and giving me all support and guidance which

made me complete the project duly. I am extremely thankful to him for providing such a nice support and guidance, although he had busy schedule managing the corporate affairs.

I owe my deep gratitude to our project guide Mr Vaibhav Diwan who take interest on my project work and guided me all along, till the completion of our project work by providing all the necessary information for developing a good project.

I would like to thank the lab staff for the operation extended to us throughout the project. After doing this project I can confidently say that this experience has not only enriched me with technical knowledge but also has unparsed the maturity of thought and vision. The attributes required in being a successful professional

Dept. of CEA, GLAU, Mathura ii

**Abstract**

Eﬃcient and accurate object detection has been an important topic in the advancement of computer vision systems. With the advent of deep learning techniques, the accuracy for object detection has increased drastically. The project aims to incorporate state-of-the-art technique for object detection with the goal of achieving high accuracy with a real-time performance. A major challenge in many of the object detection systems is the dependency on other computer vision techniques for helping the deep learning based approach, which leads to slow and non-optimal performance. In this project, we use a completely deep learning based approach to solve the problem of object detection in an end-to-end fashion. The network is trained on the most challenging publicly available dataset (PASCAL VOC), on which an object detection challenge is conducted annually. The resulting system is fast and accurate, thus aiding those applications which require object detection.

Dept. of CEA, GLAU, Mathura iii

**Contents**

Acknowledgments

Abstract

**1. Introduction**

1.1 Problem Statement ………................................................................................... 1

1.2 Application .......................................................................................................... 2

1.3 Challenges………………….................................................................................. 2

**2. Introduction to Open CV 3**

2.1 Open CV -Python...................................................................................................3

2.2 Installation of Open CV..........................................................................................4

2.3 Introduction to Numpy…………….......................................................................6

2.4 Introduction to Argparse…………………………………………………………7

**3. Introduction to YOLO Algorithm**

3.1YOLO (You Only Look Once) …………................................................................. 8

3.2Introduction to Pertained models……………………………………………………. 11

3.3Code for Object Detection………………………………………………………….. 19

**4. Conclusion**………………………………………………………………………..20

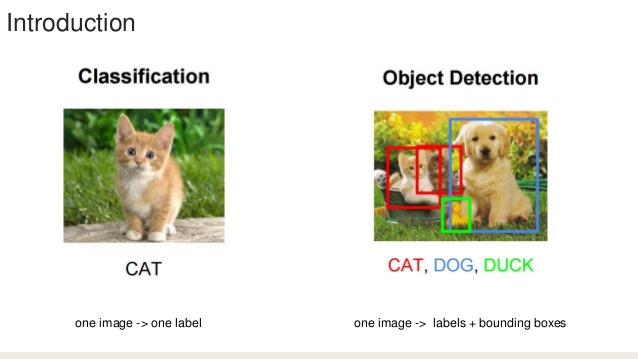
**5. Reference**………………………………………………………………………….21

**CHAPTER-1**

**Introduction**

**1.1 Problem Statement**

Many problems in computer vision were saturating on their accuracy before a decade. However, with the rise of deep learning techniques, the accuracy of these problems drastically improved. One of the major problem was that of image classiﬁcation, which is deﬁned as predicting the class of the image. A slightly complicated problem is that of image localization, where the image contains a single object and the system should predict the class of the location of the object in the image (a bounding box around the object). The more complicated problem (this project), of object detection involves both classiﬁcation and localization. In this case, the input to the system will be a image, and the output will be a bounding box corresponding to all the objects in the image, along with the class of object in each box. An overview of all these problems is depicted in Fig. 1.

 Figure 1: Computer Vision Task

Dept. of CEA, GLAU, Mathura Page 1

**1.2 Applications**

A well-known application of object detection is face detection, that is used in almost all the mobile cameras. A more generalized (multi-class) application can be used in autonomous driving where a variety of objects need to be detected. Also it has an important role to play in surveillance systems. These systems can be integrated with other tasks such as pose estimation where the ﬁrst stage in the pipeline is to detect the object, and then the second stage will be to estimate pose in the detected region. It can be used for tracking objects and thus can be used in robotics and medical applications. Thus this problem serves a multitude of applications.

The major challenge in this problem is that of the variable dimension of the output which is caused due to the variable number of objects that can be present in any given input image. Any general machine learning task requires a ﬁxed dimension of input and output for the model to be trained.

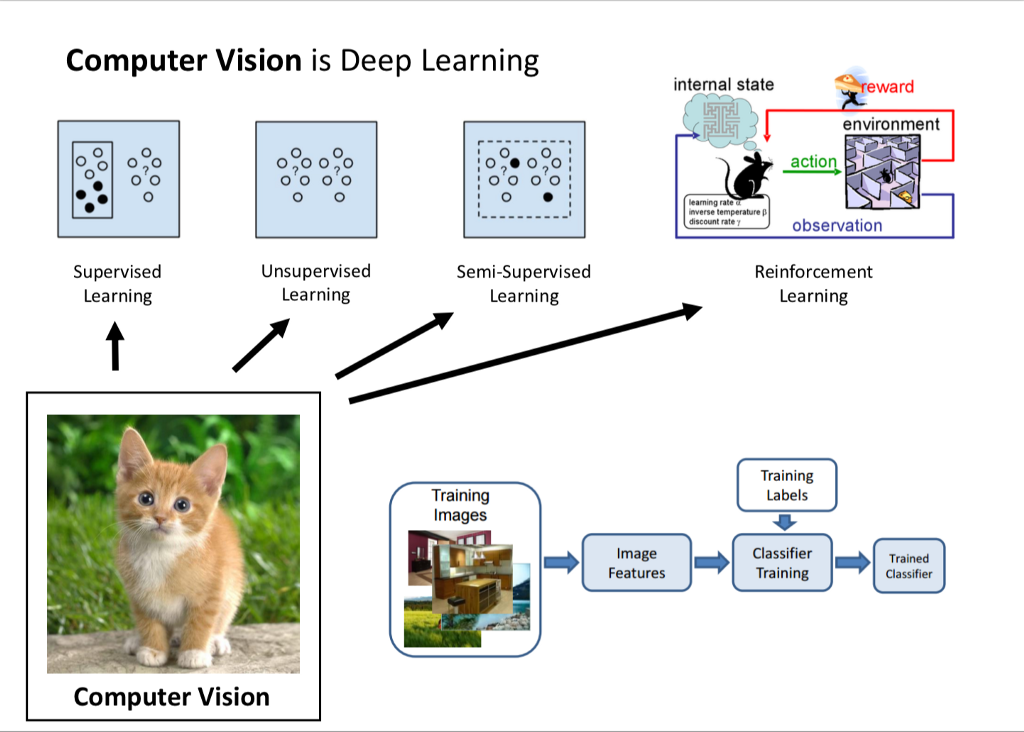
Dept. of CEA, GLAU, Mathura Page 2

**CHAPTER-2**

**INTRODUCTION TO OPEN CV WITH PYTHON**

Open CV was started at Intel in 1999 by Gary Brodsky and the first release came out in 2000.Open CV supports a lot of algorithms related to Computer Vision and Machine Learning and it is expanding day-by-day.

Currently Open CV supports a wide variety of programming languages like C++, Python, and Java etc. and is available on different platforms including Windows, Linux, OS X, Android, iOS etc. Also, interfaces based on CUDA and Open CV are also under active development for high-speed GPU operations. Open CV-Python is the Python API of Open CV. It combines the best qualities of Open CV C++ API and Python language.

 Figure 2: Computer Vision

Dept. of CEA, GLAU, Mathura Page 3

**2.1 Open CV-Python**

Python is a general purpose programming language started by Guido van Rossum, which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability. Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how Open CV-Python works it is a Python wrapper around original C++ implementation. And the support of Numpy makes the task easier. Numpy is a highly optimized library for numerical operations. It gives MATLAB-style syntax. All the Open CV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with Open CV, which increases number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this. So Open CV-Python is an appropriate tool for fast prototyping of computer vision problem. A prior knowledge on Python and Numpy is required before starting because they won’t be covered in this guide. Especially, a good knowledge on Numpy is must to write optimized codes in Open CV-Python.

Dept. of CEA, GLAU, Mathura Page 4

**2.2 Download and Installation Python**



Fig 2.2 This image shows how to install python

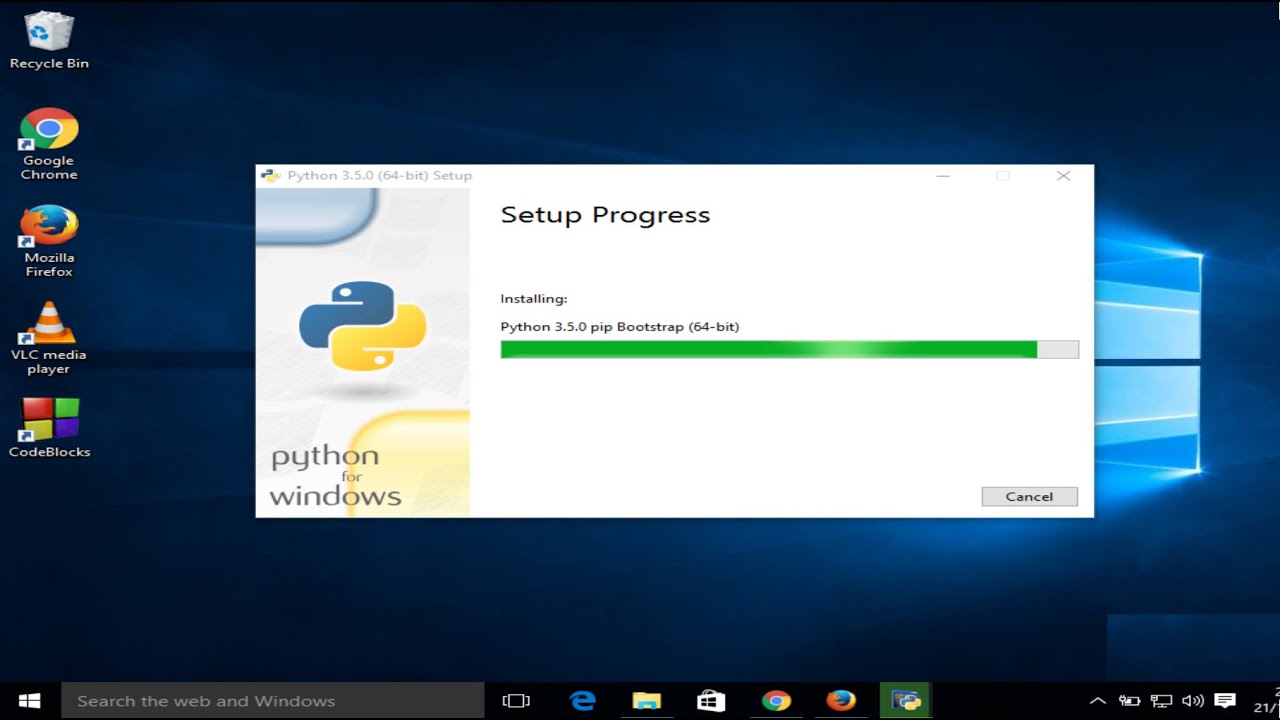


Fig 2.3 Installing python in Windows 10

Dept. of CEA, GLAU, Mathura Page 5

**2.3 Installation of Open CV**

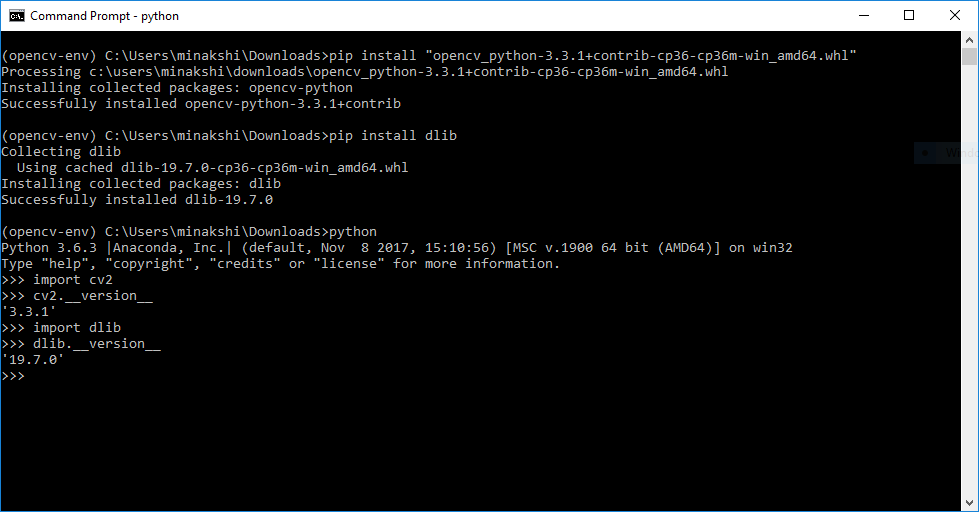


Fig 2.5 Installing open cv in cmd prompt

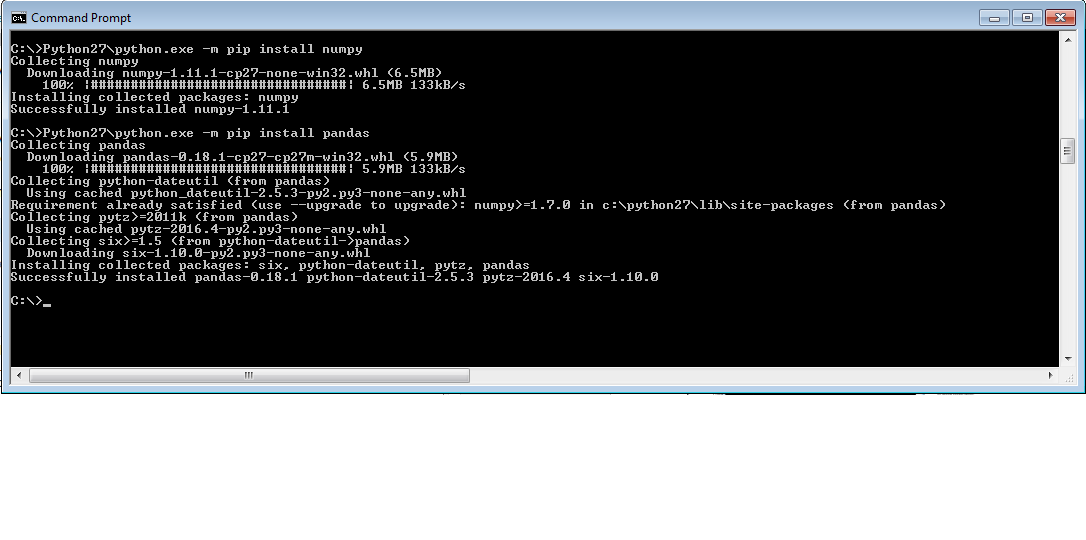
Dept. of CEA, GLAU, Mathura Page 6

**2.4 INTRODUCTION TO NUMPY**

* Numpy is an open source numerical Python library.
* Numpy contains a multi-dimensional array and matrix data structures.
* It can be utilised to perform a number of mathematical operations on arrays such as trigonometric, statistical and algebraic routines.
* Numpy is an extension of Numeric and Numarray.
* The library contains a large number of mathematical, algebraic and transformation functions.
* It also contains random number generators.
* Numpy is a wrapper around a library implemented in C

Use pip to install Numpy package:

pip install Numpy

Fig 2.6 Installing Numpy in cmd prompt

Dept. of CEA, GLAU, Mathura Page 7

**2.5 INTRODUCTION TO ARGPARSE**

Parser for command line options, arguments and subcommands.

This python module makes easy to write user-friendly command-line interfaces. The program defines what arguments it requires. Argparse will figure out how to parse those out of sys. argv.

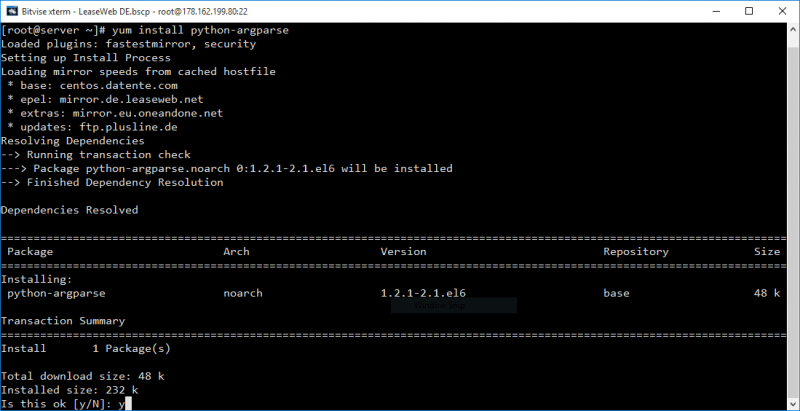


Fig 2.6 Installing Argparse in cmd prompt

Dept. of CEA, GLAU, Mathura Page 8

**CHAPTER-3**

**INTRODUCTION TO YOLO ALGORITHM**

**YOLO (You Only Look Once)**

YOLO uses deep learning and convolutional neural networks (CNN) for object detection, it stands out from its “competitors” because, as the name indicates it only needs to “see” each image once. This allows YOLO to be one of the fastest detection algorithms (naturally sacrificing some accuracy). Thanks to this swiftness YOLO can detect objects in real time (up to 30 FPS).

To carry out the detection, the image is divided in a grid of SxS (left image). Each one of the cells will predict N possible “bounding boxes” and the level of certainty (or probability) of each one of them (image at the centre), this means SxSxN boxes are calculated. The vast majority of these boxes will have a very low probability, that’s why the algorithm proceeds to delete the boxes that are below a certain threshold of minimum probability. The remaining boxes are passed through a “non-max suppression” that will eliminate possible duplicate objects and thus only leave the most exact of them (image on the right).

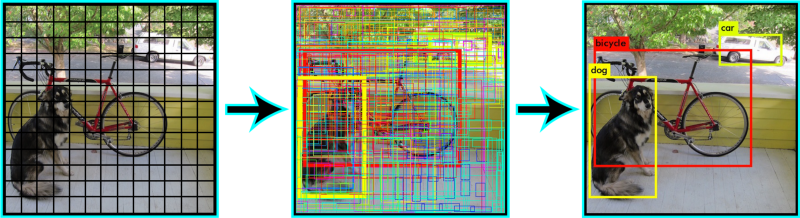


Fig 3.1 Detection of Image

Dept. of CEA, GLAU, Mathura Page 9

At the moment of writing this, YOLO has gone through three iterations, each one of them is a gradual improvement over the previous one. You can check each one of the articles

* The original. You Only Look Once: Unified, Real-Time Object Detection
* The second version. YOLO9000: Better, Faster, Stronger https:(Yes, the name is inspired by the Daft Punk song!)
* The third version. YOLOv3: An Incremental Improvement
* The algorithm applies a neural network to an entire image. The network divides the image into an S X S grid and comes up with bounding boxes, which are boxes drawn around images and predicted probabilities for each of these regions.
* The method used to come up with these probabilities is logistic regression. The bounding boxes are weighted by the associated probabilities. For class prediction, independent logistic classifiers are used.
* In this article, I am going to demonstrate how to implement the YOLO algorithm with a pre trained model.
* First, we would need to install DarkNet. DarkNet is a neural network framework that is open source. You can find more information about DarkNet

Dept. of CEA, GLAU, Mathura Page 10

# **Chapter 3.2**

# **Introduction to per trained Models**

**Select a pre-trained model.**

From the wide range of pre-trained models that are available, you pick one that looks suitable for your problem

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

Dept. of CEA, GLAU, Mathura Page 11

**Chapter 3.3**

**Code for Object Detection**

**#import the necessary libraries**

**# computer vision library**

import cv2

**# argparse use to give the runtime argument**

import argparse

**# to use numerical function**

import numpy as np

**#Construct the argument parse and parse the arguments**

ap = argparse.ArgumentParser()

ap.add\_argument ('-i', '--image', required=True, help = 'path to input image')

ap.add\_argument('-c', '--config', required=True,

help = 'path to yolo config file')

ap.add\_argument('-w', '--weights', required=True,

help = 'path to yolo pre-trained weights')

ap.add\_argument('-cl', '--classes', required=True,

help = 'path to text file containing class names')

args = ap. parse\_args()

Dept. of CEA, GLAU, Mathura Page 13

**#function to get the output layer names in architecture**

def get\_output\_layers(net):

layer\_names = net.getLayerNames()

output\_layers = [layer\_names [i [0] - 1] for i in net.getUnconnectedOutLayers()]

return output\_layers

**# fuction to draw bounding box on the detected object with class name**

def draw\_prediction(img, class\_id, confidence, x, y, x\_plus\_w, y\_plus\_h):

label = str(classes[class\_id])

color = COLORS[class\_id]

cv2.rectangle(img, (x,y), (x\_plus\_w,y\_plus\_h), color, 2)

cv2.putText(img, label, (x-10,y-10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, color, 2)

image = cv2.imread(args.image)

Width = image.shape[1]

Height = image.shape[0]

scale = 0.00392

Dept. of CEA, GLAU, Mathura Page 14

**#read class names from text file**

classes = None

with open(args.classes, 'r') as f:

classes = [line.strip() for line in f.readlines()]

**#generate different colors for different classes**

COLORS = np.random.uniform(0, 255, size=(len(classes), 3))

**#read pre-trained model and config** file

net = cv2.dnn.readNet(args. weights, args.config)

**#create input blob (binary large object)**

blob = cv2.dnn.blobFromImage(image, scale, (416,416), (0,0,0), True, crop=False)

**#set input blob for the network**

net.setInput(blob)

**# run inference through the network and gather predictions from output layer**

outs = net.forward(get\_output\_layers(net))

Dept. of CEA, GLAU, Mathura Page 15

**#initializations**

class\_ids = []

confidences = []

boxes = []

conf \_threshold = 0.5

nms\_threshold = 0.4

**#for each detection from each output layer**

**#get the confidence ,class id, bounding box params**

**#and ignore weak detections (confidence < 0.5)**

for out in outs:

for detection in out:

scores = detection[5:]

class\_id = np.argmax(scores)

confidence = scores[class\_id]

if confidence > 0.5:

center\_x = int(detection[0] \* Width)

center\_y = int(detection[1] \* Height)

w = int(detection[2] \* Width

h = int(detection[3] \* Height)

Dept. of CEA, GLAU, Mathura Page 16

x = center\_x - w / 2

y = center\_y - h / 2

class\_ids.append(class\_id)

confidences.append(float(confidence))

boxes.append([x, y, w, h])

**#apply non-max suppression**

indices = cv2.dnn.NMSBoxes(boxes, confidences, conf\_threshold, nms\_threshold)

**# go through the detections remaining**

**# after nms and draw bounding box**

for i in indices:

i = i[0]

box = boxes[i]

x = box[0]

y = box[1]

w = box[2]

h = box[3]

draw\_prediction(image, class\_ids[i], confidences[i], round(x), round(y), round(x+w), round(y+h))

Dept. of CEA, GLAU, Mathura Page 17

**# display output image**

**#cv2.imshow(out\_image\_name, image)**

cv2.imshow("object detection", image)

**# wait until any key is pressed**

cv2.waitKey()

cv2.imwrite("object-detection.jpg", image)

cv2.destroyAllWindows()

**Output:**

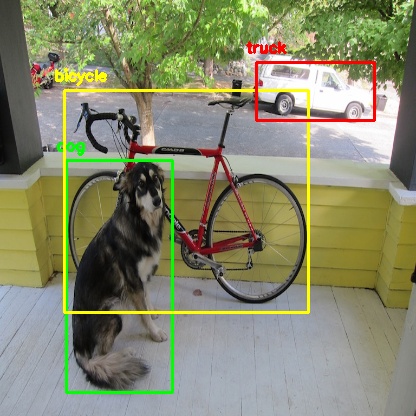


Fig 3.2: Outputs of Object Detection

Dept. of CEA, GLAU, Mathura Page 18

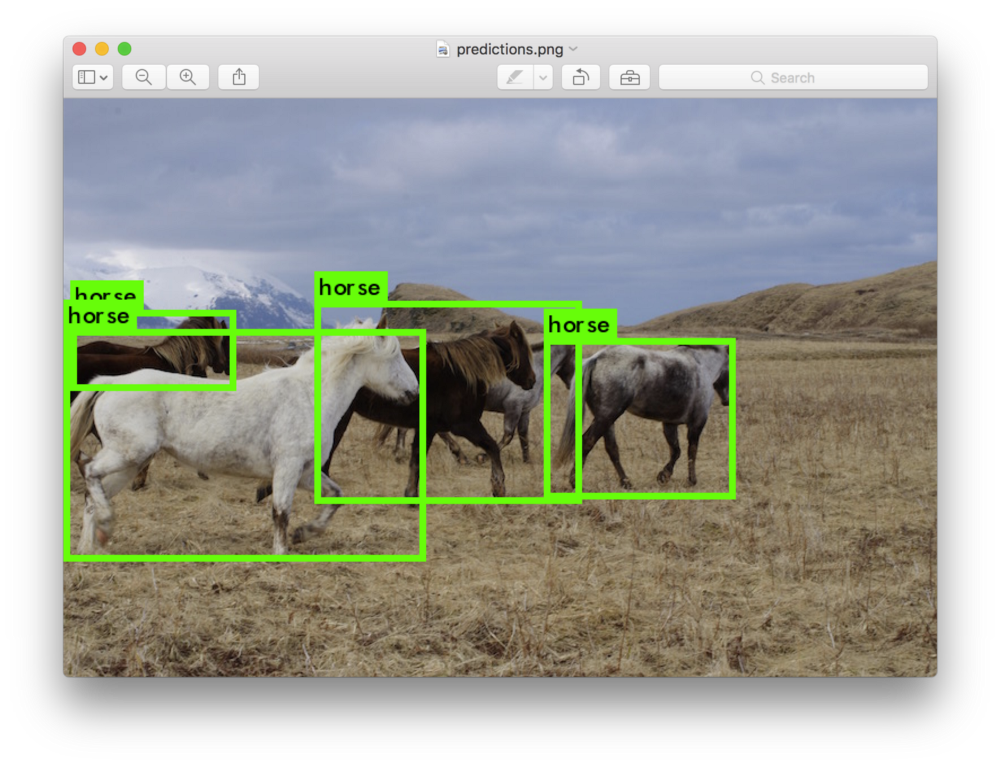


Fig 3.3: Outputs of Object Detection

Dept. of CEA, GLAU, Mathura Page 19

**Chapter 4**

**Conclusion**

An accurate and eﬃcient object detection system has been developed which achieves comparable metrics with the existing state-of-the-art system. This project uses recent techniques in the ﬁeld of computer vision and deep learning. Custom dataset was created using labelling and the evaluation was consistent. This can be used in real-time applications which require object detection for pre-processing in their pipeline. An important scope would be to train the system on a video sequence for usage in tracking applications. Addition of a temporally consistent network would enable smooth detection and more optimal than per-frame detection.

Dept. of CEA, GLAU, Mathura Page 20

**Chapter 5**

# **References:**

<https://docs.python.org/dev/library/argparse.html>

[www.google.com](http://www.google.com)

[www.edureka.com](http://www.edureka.com)

Dept. of CEA, GLAU, Mathura Page 21