

A

Mini-Project II Report On

“Real -Time Object Detection”

Submitted In Partial Fulfilment of the Requirement

For The Award of Third Year Bachelor of Technology

In Computer Science and Engineering of

Dr. Babasaheb Ambedkar Technological University, Lonere.

Submitted By

Mr. Patil Nitish Kailas

PRN Number: 2151701242003

Under The Guidance of

Dr. D. D. Patil



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SHRI SANT GADGE BABA

COLLEGE OF ENGINEERING AND TECHNOLOGY, BHUSAWAL - 425203

DR. BABASAHEB AMBEDKAR TECHNOLOGICAL UNIVERSITY, LONERE.

2023-2024

**Shri Sant Gadge Baba
College of Engineering and Technology,
Bhusawal 425 203**



Certificate

This is to certify that **Mr. Patil Nitish Kailas** has successfully completed her Mini-project II on “**Real-Time Object Detection**” for the partial fulfilment of the award of Third Year of Engineering in the Computer Science & Engineering as prescribed by the Dr. Babasaheb Ambedkar Technological University, Lonere during academic year 2023-2024

Dr. D. D. Patil
(Guide)

Dr. D. D. Patil
(H.O.D.)

Dr. R. B. Barjibhe
(Principal)

D E C L A R A T I O N

I hereby declare that the Mini-Project Report entitled, “**Real-Time Object Detection**” is written and studied by me under the guidance of **Dr. D. D. Patil H.O.D** of Department of Computer Science and Engineering, Shri Sant Gadge Baba College of Engineering and Technology, Bhusawal. This report is written by studying the various articles, books, papers, journals, and other resources available on internet out of which some of them are listed in the end of report.

Place: Bhusawal

Date:

Signature Of Student

Nitish Kailas Patil

PRN No:2151701242003

SYNOPSIS

An object detection system recognises and searches the objects of the real world out of a digital image or a video, where the object can belong to any class or category, for example humans, cars, vehicles and so on. Authors have used OpenCV packages, Caffe model, Python and NumPy to complete this task of detecting an object in an image or a video. This review paper discusses how deep learning technique is used to detect a live object, localise an object, categorise an object, extract features, appearance information, and many more, in images and videos using OpenCV and how caffe model is used to implement and why authors have chosen caffe model over other frameworks. To build our deep learning-based real-time object detector with OpenCV we need to access webcam in an efficient manner and to apply object detection to each frame.

Recognition of handwritten mathematical expressions is helpful in writing technical documents as well as useful in converting handwritten documents with mathematical equations into electronic format. Symbol recognition in mathematical expressions is a complex task due to large character set and writer variability in size and style of symbols. In this project an assistance system is developed that analyses the handwritten mathematical equations. In this work, mathematical expression recognition task is carried out in different phases which include data collection, preprocessing, segmentation, feature extraction, symbol classification as well as mathematical expression. The system will be able to recognize images of handwritten equations and output the result of the equation. Being able to recognize handwritten equations, it has applications for consumers and academics.

Furthermore, whereas handwritten equations are human readable, the “code” of typesetting languages are often highly nested and difficult to edit. These problems of typesetting present an opportunity for improving the workflow of writing math equations digitally. Handwritten character or symbol recognition is one of the applications in pattern classification. It is generally easy for anyone to recognize handwritten or characters and symbols but it is difficult for a computer to recognize them. This difficulty can be overcome by adopting machine learning approach by designing a system that recognizes the patterns.

Keywords: - segmentation, feature extraction, symbol classification, machine learning, typesetting, Object Detection, OpenCV, Images, Videos, Caffe model.

I N D E X

Chapter No.	Title	Page No
	Title Sheet	i
	Certificate	ii
	Declaration	iii
	Synopsis	iv
	Index	v
	List of Figures	vii
1	Introduction	1
2	Problem identification and Project objectives	2
	2.1 Problem Statement for Real-time Object Detection.	2
3	Literature Survey	3
	3.1 Real-time Object Detection.	3
	3.1.1 Earlier System.	3
	3.1.2 Problem associated with earlier system.	3
	3.1.3 Proposed System.	3
4	Analysis	5
	4.1 Analysis of Real-time Object Detection	5
5	Design Methodology	8
	5.1 Design Methodology Real-Time Object Detection.	8
	5.1.1 Algorithm.	9
	5.1.2 Techniques used.	13
	5.2 Design Methodology Handwritten Equation Solver.	15
6	Flow Chart And Coding	17
7	Running screenshot of project	21
8	Application	22
	8.1 Application of Real-time Object Detection.	22
9	Conclusion	23
	Acknowledgement	24
	References	25

LIST OF FIGURES

Figure No.	Title of Figure
2.1	Computer Vision task
3.1	Block diagram of real-time Object detection
5.1	Data flow diagram of real time object detection
5.2	Jaccard distance
5.3	Architecture overview
5.4	Two stage method
5.5	Unified model
5.6	E-R Diagram of Real-time objects detection.
5.7	Use case Diagram of Real-time object detection.
5.8	Use case diagram
5.9	DFD (level 0) for HES system
5.10	DFD (level 1) for image reader
5.11	DFD (level 1) for recognition unit
5.12	DFD for CNN
6.1	Realtime object detection flowchart
7.1.1	Loading model and initializing video stream
7.1.2	Initializing webcam
7.1.3	Realtime objection detection result

CHAPTER 1

INTRODUCTION

1.1 Introduction of Real-time Object Detection:

Object Detection is the process of finding and recognizing real-world object instances such as car, bike, TV, flowers, and humans out of an images or videos. An object detection technique lets you understand the details of an image or a video as it allows for the recognition, localization, and detection of multiple objects within an image [16]. It is usually utilized in applications like image retrieval, security, surveillance, and advanced driver assistance systems (ADAS). Object Detection is done through many ways:

- Feature Based Object Detection
- Viola Jones Object Detection
- SVM Classifications with HOG Features
- Deep Learning Object Detection

Object detection from a video in video surveillance applications is the major task these days. Object detection technique is used to identify required objects in video sequences and to cluster pixels of these objects.

Deep learning has gained a tremendous influence on how the world is adapting to Artificial Intelligence since past few years. Some of the popular object detection algorithms are Region-based Convolutional Neural Networks (RCNN), Faster- RCNN, Single Shot Detector (SSD) and You Only Look Once (YOLO). Amongst these, Faster CNN and SSD have better accuracy, while YOLO performs better when speed is given preference over accuracy.

Deep learning combines SSD and Mobile Nets to perform efficient implementation of detection and tracking. This algorithm performs efficient object detection while not compromising on the performance.

The detection of an object in video sequence plays a major role in several applications specifically as video surveillance applications.

Object detection in a video stream can be done by processes like pre-processing, segmentation, foreground and background extraction, feature extraction.

CHAPTER 2

PROBLEM STATEMENT

2.1. Problem Statement for Real-time Object Detection.

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 problems was that of image classification, which is defined as Predicting the class of the image. In this case, the input to the system will be an 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.2.1.

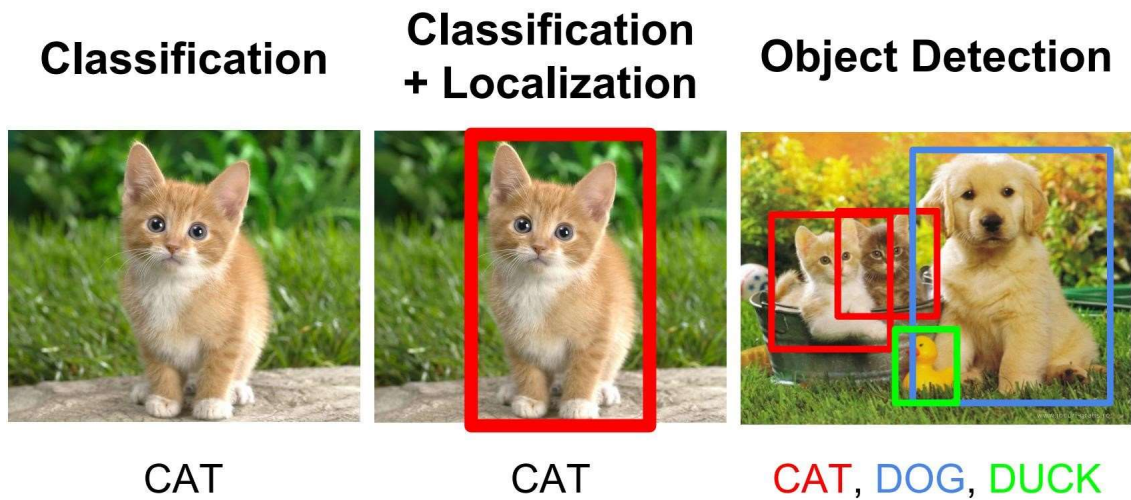


figure 2.1: Computer Vision Task.

This application is useful for recognizing mathematical expression given as an input image once input image is given to propose system then it will calculate result of given expression in the image. Recognition and classification are done by neural network.

CHAPTER 3

LITERATURE SURVEY

3.1 Real-time Object Detection.

3.1.1. Earlier System.

In the late 1960s, computer vision began at universities which were pioneering artificial intelligence. It was meant to mimic the human visual system, as a stepping stone to endowing robots with intelligent behavior. In 1966, it was believed that this could be achieved through a summer project, by attaching a camera to a computer and having it "describe what it saw". What distinguished computer vision from the prevalent field of digital image processing at that time was a desire to extract three-dimensional structure from images with the goal of achieving full scene understanding. Studies in the 1970s formed the early foundations for many of the computer vision algorithms that exist today, including extraction of edges from images, labelling of lines, non-polyhedral and polyhedral modelling, representation of objects as interconnections of smaller structures, optical flow, and motion estimation.

3.1.2. Problem associated with earlier system.

Before the emergence of machine learning and deep learning, computer vision was just an assumption in that era. There were many problems that lack computer vision to exist, which are as follows:-

- a) Lack of hardware support.
- b) Lack of hardware power.
- c) Absence of deep learning models and algorithms.
- d) Languages support.
- e) Lack of mathematical and statistical tools.

3.1.3. Proposed System.

Earlier system comes up with many flaws, lack of resources and lack of software and hardware support, to overcome with this obstacle we proposed a real time detection model which have advanced deep learning model, statistical tools, languages support (python in our case), advanced GPU to render images sharply, Webcam to fetch live frames (images) for live Detection and a beautiful computer vision library like OpenCV.

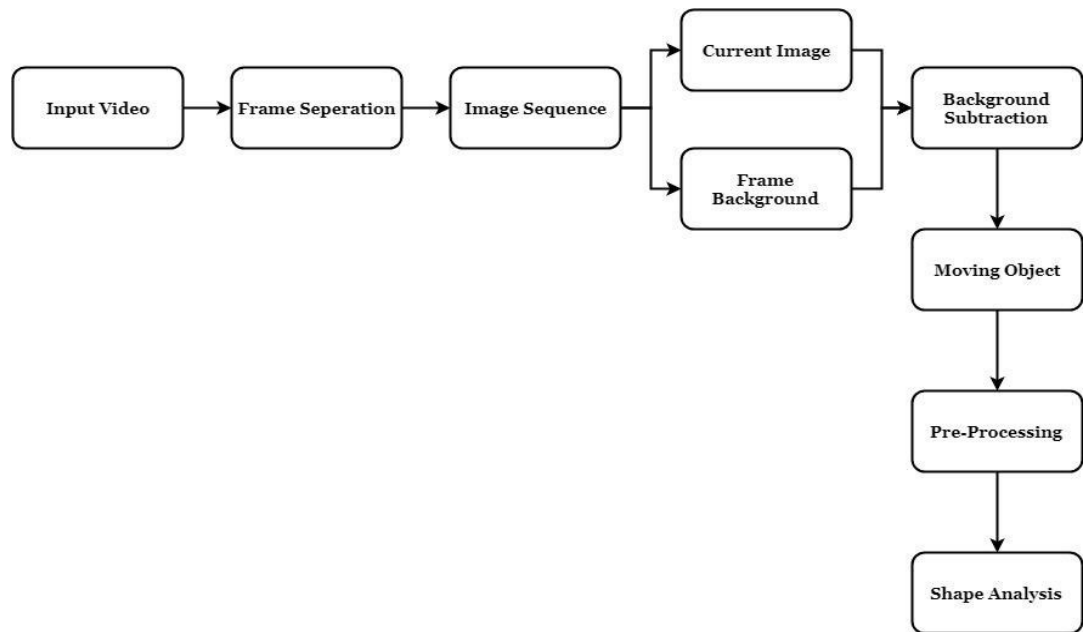


Fig3.1: -Block diagram of Real-time Object Detection.

CHAPTER 4

ANALYSIS

REQUIREMENT ANALYSIS

4.1. Real-time Object Detection

Hardware Requirement

- ☐ Webcam.



4 GB ram or more (Recommended).



- ☐ Graphics Card (Minimum 2GB Recommended).



Software Requirement

- ☐ Python 3 (version 3.7 recommended).
- ☐ MobilenetSSD (Deep learning neural network).
- ☐ Better IDE like Jupiter notebook, PyCharm.



REQUIREMENT SPECIFICATION

a) FUNCTIONAL REQUIREMENTS

The functional requirements for a system describe what system do.

1. The developed system should recognize handwritten mathematical expression present in the image.
2. System must provide the quality of service to user.
3. System must provide accuracy for character recognition.

NORMAL REQUIREMENTS

These are the requirements clearly stated by the customer hence requirement must be present for customer satisfaction.

- **N1:** Application should have graphical user interface.
- **N2:** Input of characters with various font size and styles should recognize.
- **N3:** Application should be able for matching the stored patterns on input handwritten character

VALIDATION OF REQUIREMENTS

The project “HES using Neural Network” will be recognized as successful implementation if it provide all the required images on the basis of suitable input with minimum time. The requirement specification define should be validated such a that the successful implementation of product can be recognized. Hence validation specifies classes of tests to be performed to validate function, performance and the constraints. With respect to the system under consideration the following issues are to be validated to ensure consistency of system.

- **V1:** The Neural Network are known to be capable of providing good recognition rate at the present as compare to other methods.
- **V2:** Handwritten Mathematical Expression Recognition system give much better result in terms of performance and accuracy in comparison with existing usual approach due to the application of artificial way character recognition and Neural Network in detection of characters.
- **V3:** Handwritten Mathematical Expression Recognition technology provides image definition, image preprocessing and image segmentation and recognition capabilities and still maintains high level of accuracy.

SYSTEM REQUIREMENTS

Hardware Requirements

- Intel i3 Processer
- 128 MB RAM
- 10 GB Hard Disk.

Software Requirements

- Windows 7/8/8.1
- Language: Python(including keras, numpy, pandas, pickle).
- Eclipse

CHAPTER 5

DESIGN METHODOLOGY

5.1. Real-Time Object Detection.

There has been a lot of work in object detection using traditional computer vision techniques (sliding windows, deformable part models). However, they lack the accuracy of deep learning-based techniques. Among the deep learning-based techniques, two broad class of methods are prevalent: two stage detection (RCNN [1], Fast RCNN [2], Faster RCNN [3]) and unified Detection (Yolo [4], SSD [5]). The major concepts involved in these techniques have been Explained below.

DATA FLOW DIAGRAM:

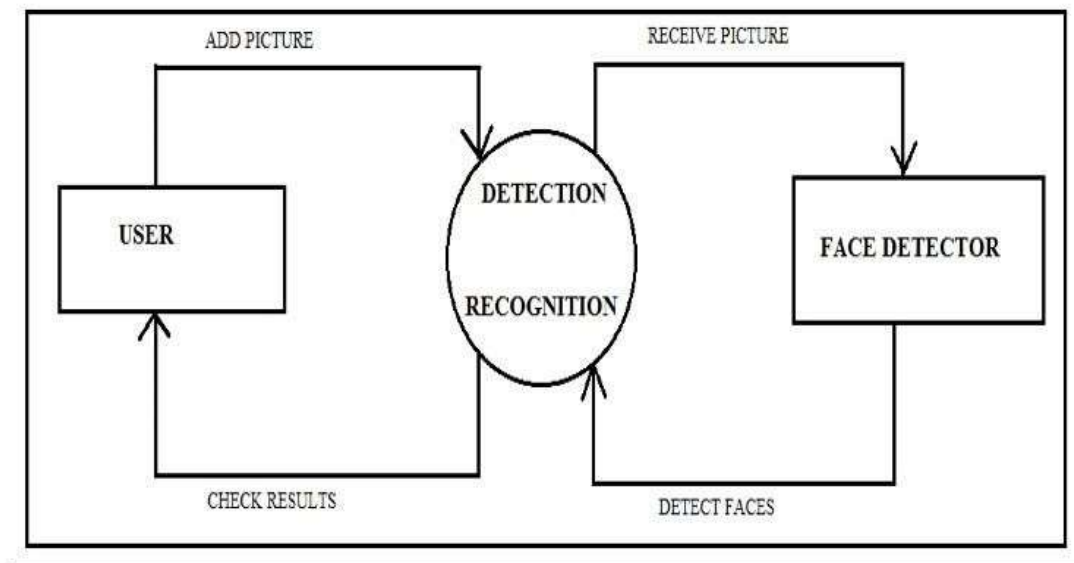


Fig 5.1:-Data Flow Diagram for Real-Time Object Detection.

In the above data flow diagram, we have shown the the two entities that are linked with specific process. The arrows are the dependency associated either with entities to process or process to entities.

Entities are user and face detector, which is linked with the process of detection recognition by a means of note or an dependency arrow.

Types of link associated are Add picture, Receive Picture, Detect faces, Check result. The major concepts involved in these techniques have been explained below.

a) Bounding Box.

The bounding box is a rectangle drawn on the image which tightly fits the object in the image. A bounding box exists for every instance of every object in the image. For the box, 4 numbers (centre x, centre y, width, height) are predicted. This can be trained using a distance measure between predicted and ground truth bounding box. The distance measure is a Jaccard distance which computes intersection over union between the predicted and ground truth boxes as shown in Fig. 5.2.

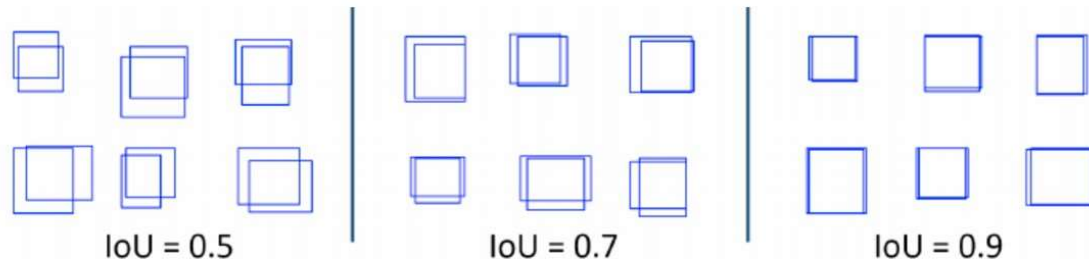


Fig 5.2: - Jaccard distance

b) Classification + Regression.

The bounding box is predicted using regression and the class within the bounding box is predicted using classification. The overview of the architecture is shown in Fig. 4

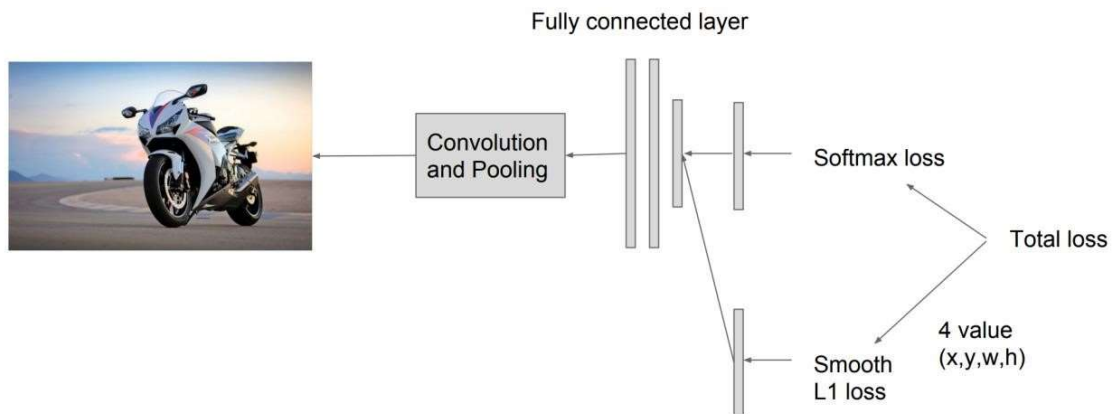
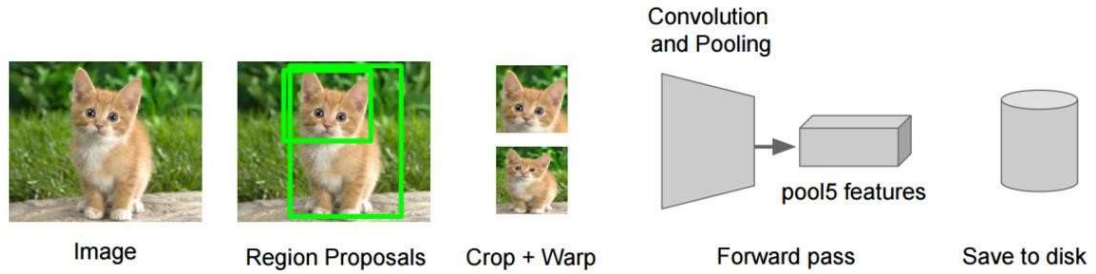


Fig 5.3: - Architecture overview

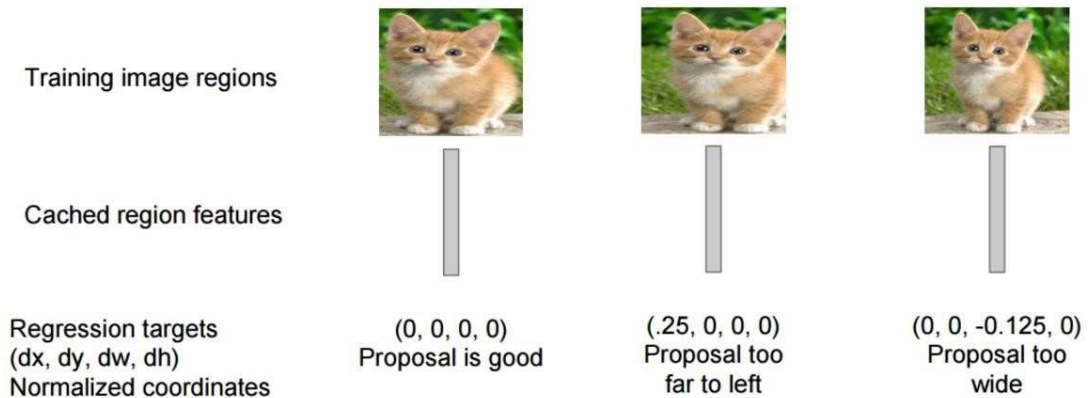
c) Two-stage Method.

In this case, the proposals are extracted using some other computer vision technique and then resized to fixed input for the classification network, which acts as a feature extractor.

Then an SVM is trained to classify between object and background (one SVM for each class). Also a bounding box regressor is trained that outputs some some correction (o_sets) for proposal boxes. The overall idea is shown in Fig. 5 These methods are very accurate but are computationally intensive (low fps).



(a) Stage one.



(b) Stage two.

Fig 5.4: -Two Stage method.

d) Unified Method.

The difference here is that instead of producing proposals, pre-define a set of boxes to look for objects. Using convolutional feature maps from later layers of the network, run another network over these feature maps to predict class scores and bounding box onsets. The broad idea is depicted in Fig. 6. The steps are mentioned below: 1. Train a CNN with regression and classification objective.

2. Gather activation from later layers to infer classification and location with a fully connected or convolutional layers.
3. During training, use Jaccard distance to relate predictions with the ground truth.
4. During inference, use non-maxima suppression to later multiple boxes around the same object.

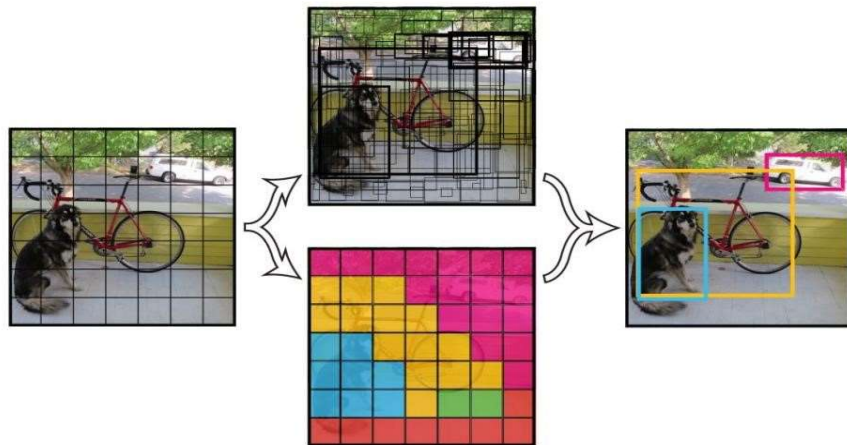


Fig 5.5: - Unified Method.

The major techniques that follow this strategy are: SSD (uses different activation maps (multiple-scales) for prediction of classes and bounding boxes) and Yolo (uses a single activation map for prediction of classes and bounding boxes). Using multiple scales helps to achieve a higher mAP(mean average precision) by being able to detect objects with different sizes on the image better. Thus the technique used in this project is SSD.

ENTITY RELATIONSHIP DIAGRAM:

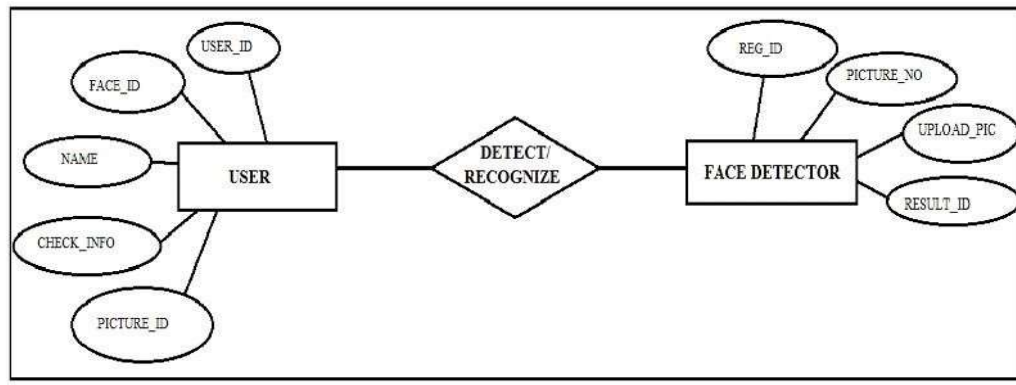


Fig 5.6: -E-R Diagram of Real-time objects detection.

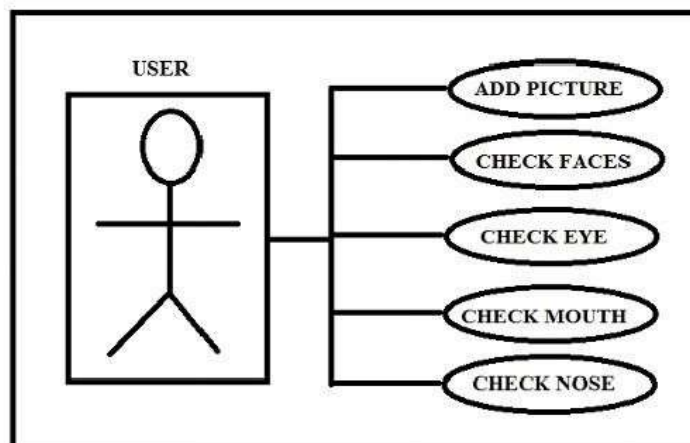
Entity relationship diagram displays the relationships of entity set stored in a database. In other words, we can say that ER diagrams help you to explain the logical structure of databases. In the above case, there are two strong Entities: -

- a) USER.
- b) FACE DETECTOR.

There is also one process present in between two entities which is Detect/Recognize. Also there is many to one relationship between attributes with entities, attributes are as follows:
-USER_ID, FACE_ID, REG_ID, PICTURE_ID etc.

USE CASE DIAGRAM:

User Module:



Software Module:

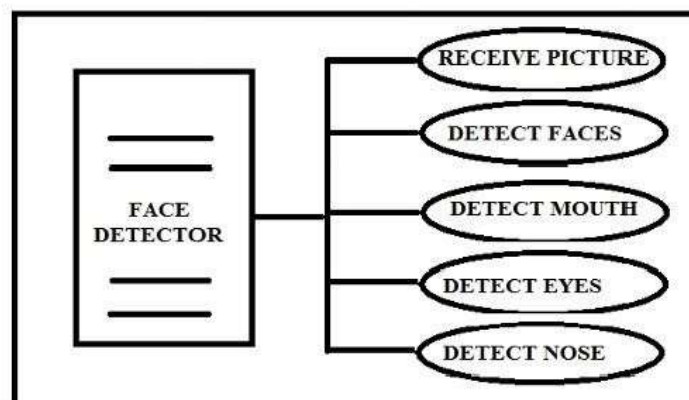


Fig 5.7:-Use case Diagram of Real-time object detection.

A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram.).

In above case there are two actors that are shown in above diagrams, they are:- A)

User and Face detectors.

B) There are two boundary of system is present which are User module and Software Module.

C) Link is associated from actor to use cases act as a relationship.

Use cases are: - add picture, receive picture, detect noise etc.

5.1.1. Techniques used.

5.1.1.1 Deep learning

The field of artificial intelligence is essential when machines can do tasks that typically require human intelligence. It comes under the layer of machine learning, where machines can acquire skills and learn from past experience without any involvement of human. Deep learning comes under machine learning where artificial neural networks, algorithms inspired by the human brain, learn from large amounts of data. The concept of deep learning is based on humans' experiences; the deep learning algorithm would perform a task continuously so that it can improve the outcome. Neural networks have various (deep) layers that enable learning. Any drawback that needs "thought" to work out could be a drawback deep learning can learn to unravel.

5.1.1.2 OpenCV

OpenCV stands for Open source Computer Vision Library is an open source computer vision and machine learning software system library. The purpose of creation of OpenCV was to produce a standard infrastructure for computer vision applications and to accelerate the utilization of machine perception within the business product [6]. It is a rich wholesome library as it contains 2500 optimized algorithms, which also includes a comprehensive set of both classic and progressive computer vision and machine learning algorithms. These algorithms are used for various functions such as discover and acknowledging faces. Identify objects classify human actions. In videos, track camera movements, track moving objects.

5.1.1.3 Caffe Model

Caffe is a framework of Deep Learning and it was made used for the implementation and to access the following things in an object detection system.

- Expression: Models and optimizations are defined as plaintext schemas in the caffe model unlike others which use codes for this purpose.
- Speed: for research and industry alike speed is crucial for state-of-the-art models and massive data [11].
- Modularity: Flexibility and extension is majorly required for the new tasks and different settings
- Openness: Common code, reference models, and reproducibility are the basic requirements of scientific and applied progress.

ANALYSIS MODEL

Behavioural Model

Use case Diagram –

The use case view models functionality of the system as perceived by outside uses. A use case is a coherent unit of functionality expressed as a transaction among actors and the system.

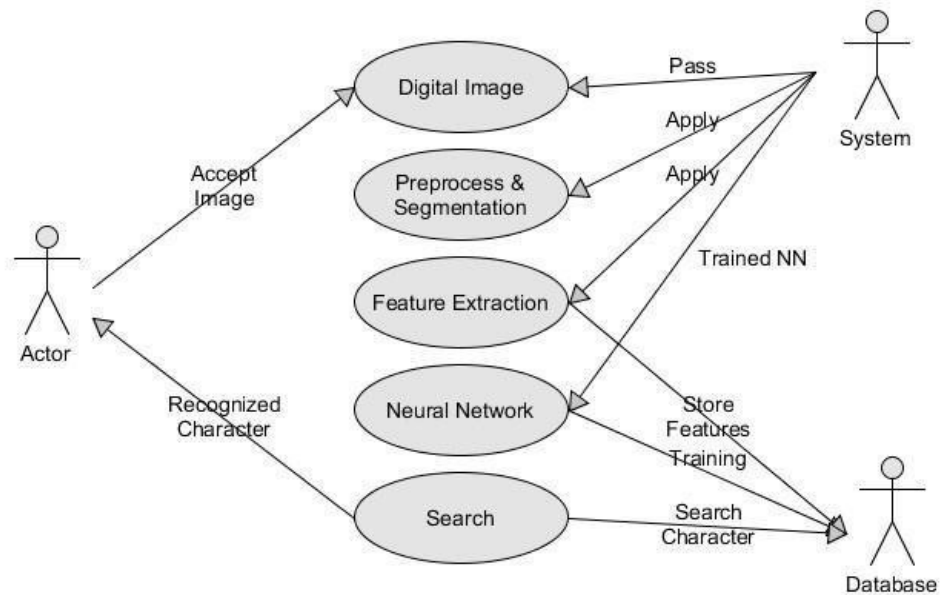


Fig 5.8: Use Case Diagram

FUNCTIONAL MODELLING

Data Flow Diagram

Data flow diagram (DFD) is also called as Bubble Chart is a graphical technique, which is used to represent information flow, and transformers those are applied when data moves from input to output. DFD represents system requirements clearly and identify transformers those becomes programs in design. DFD may further partitioned into different levels to show detailed information flow e.g. level 0, level 1 etc.



Fig 5.9: DFD (level 0) for HES System

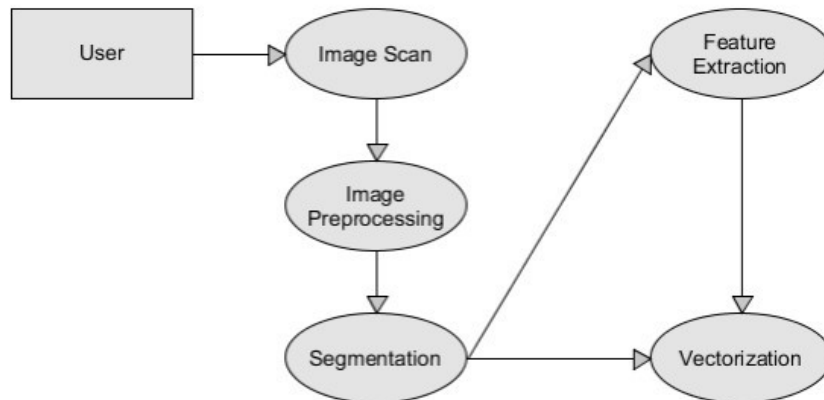


Fig 5.10: DFD (level 1) for Image Reader

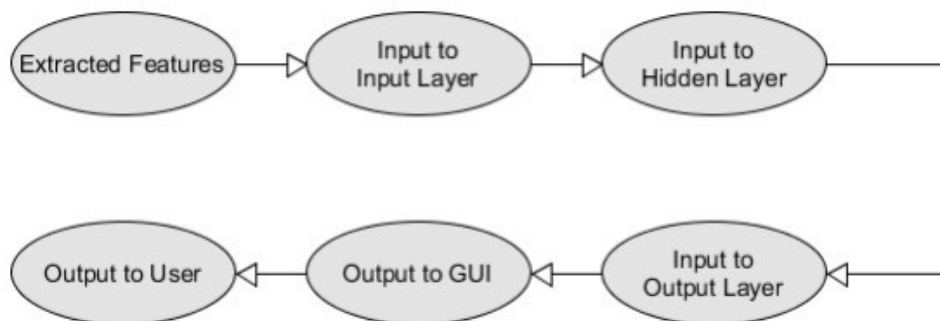


Fig 5.11: DFD (level 1) for Recognition Unit

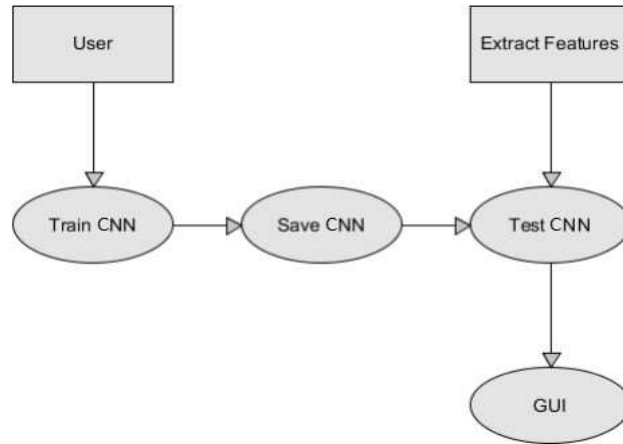


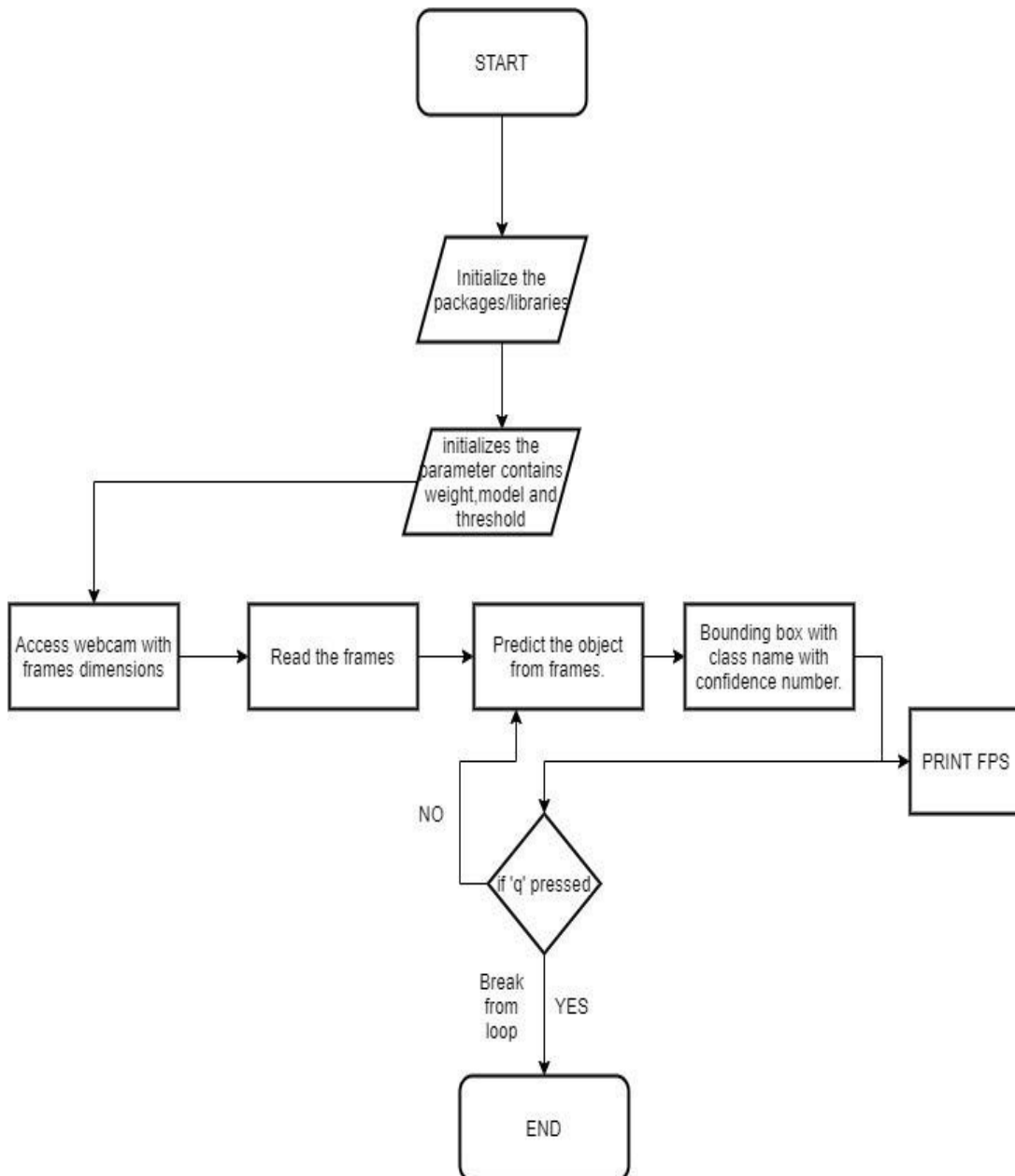
Fig 5.12: Data Flow Diagram for CNN

CHAPTER 6

FLOW-CHART & CODING

6.1 REALTIME OBJECT DETECTION.

Flowchart



Program Explanation:- Step1:

```
1. | # import the necessary packages
2. | from imutils.video import VideoStream
3. | from imutils.video import FPS
4. | import numpy as np
5. | import argparse
6. | import imutils
7. | import time
8. | import cv2
```

We begin by importing packages on **Lines 2-8**. It requires `imutils` and `opencv3.3` libraries along with `numpy`, `argparse`.

Step2:

```
10. | # construct the argument parse and parse the arguments
11. | ap = argparse.ArgumentParser()
12. | ap.add_argument("-p", "--prototxt", required=True,
13. |     help="path to Caffe 'deploy' prototxt file")
14. | ap.add_argument("-m", "--model", required=True,
15. |     help="path to Caffe pre-trained model")
16. | ap.add_argument("-c", "--confidence", type=float, default=0.2,
17. |     help="minimum probability to filter weak detections")
18. | args = vars(ap.parse_args())
```

We'll parse our command line argument.

Step3:

```
20. | # initialize the list of class labels MobileNet SSD was trained to
21. | # detect, then generate a set of bounding box colors for each class
22. | CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
23. |     "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
24. |     "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
25. |     "sofa", "train", "tvmonitor"]
26. | COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))
```

On **Lines 22-26** we initialize CLASS labels and corresponding random COLORS.

Step4:

```
28. | # load our serialized model from disk
29. | print("[INFO] loading model...")
30. | net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
31. |
32. | # initialize the video stream, allow the camera sensor to warmup,
33. | # and initialize the FPS counter
34. | print("[INFO] starting video stream...")
35. | vs = VideoStream(src=0).start()
36. | time.sleep(2.0)
37. | fps = FPS().start()
```

We load our serialized model, providing the references to our `prototxt` and `model` files on **Line 30** — notice how easy this is in `OpenCV 3.3`. Next let's initialize our video stream (this can be from a video file or a camera). First we start the `VideoStream` (**Line 35**), then we wait for the camera to warm up (**Line 36**), and finally we start the frames per second counter (**Line 37**). The `VideoStream` and `FPS` classes are part of my `imutils` package.

Step5:

```
39. # loop over the frames from the video stream
40. while True:
41.     # grab the frame from the threaded video stream and resize it
42.     # to have a maximum width of 400 pixels
43.     frame = vs.read()
44.     frame = imutils.resize(frame, width=400)
45.
46.     # grab the frame dimensions and convert it to a blob
47.     (h, w) = frame.shape[:2]
48.     blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),
49.                                   0.007843, (300, 300), 127.5)
50.
51.     # pass the blob through the network and obtain the detections and
52.     # predictions
53.     net.setInput(blob)
54.     detections = net.forward()
```

First, we read a frame (**Line 43**) from the stream, followed by resizing it (**Line 44**). Since we will need the width and height later, we grab these on **Line 47**. This is followed by converting the frame to a blob with the dnn module (**Lines 48 and 49**). Now for the heavy lifting: we set the blob as the input to our neural network (**Line 53**) and feed the input through the net (**Line 54**) which gives us our detections.

Step6:

```
55. # loop over the detections
56. for i in np.arange(0, detections.shape[2]):
57.     # extract the confidence (i.e., probability) associated with
58.     # the prediction
59.     confidence = detections[0, 0, i, 2]
60.
61.     # filter out weak detections by ensuring the 'confidence' is
62.     # greater than the minimum confidence
63.     if confidence > args["confidence"]:
64.         # extract the index of the class label from the
65.         # 'detections', then compute the (x, y)-coordinates of
66.         # the bounding box for the object
67.         idx = int(detections[0, 0, i, 1])
68.         box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
69.         (startX, startY, endX, endY) = box.astype("int")
70.
71.         # draw the prediction on the frame
72.         label = "{}: {:.2f}%".format(CLASSES[idx],
73.                                     confidence * 100)
74.         cv2.rectangle(frame, (startX, startY), (endX, endY),
75.                       COLORS[idx], 2)
76.         y = startY - 15 if startY - 15 > 15 else startY + 15
77.         cv2.putText(frame, label, (startX, y),
78.                     cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)
```

We start by looping over our Detection, keeping in mind that multiple objects can be detected in a single image. We also apply a check to the confidence (i.e., probability) associated with each detection. If the confidence is high enough (i.e. above the threshold), then we'll display the prediction in the terminal as well as draw the prediction on the image with text and a colored bounding box.

Step7:

```
80. | # show the output frame
81. | cv2.imshow("Frame", frame)
82. | key = cv2.waitKey(1) & 0xFF
83. |
84. | # if the `q` key was pressed, break from the loop
85. | if key == ord("q"):
86. |     break
87. |
88. | # update the FPS counter
89. | fps.update()
```

The above code block is pretty self-explanatory — first we display the frame (**Line 82**). Then we capture a key press (**Line 83**) while checking if the ‘q’ key (for “quit”) is pressed, at which point we break out of the frame capture loop (**Lines 86 and 87**). Finally we update our fps counter (**Line 90**).

Step8:

```
91. | # stop the timer and display FPS information
92. | fps.stop()
93. | print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
94. | print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
95. |
96. | # do a bit of cleanup
97. | cv2.destroyAllWindows()
98. | vs.stop()
```

When we’ve exited the loop, we stop the fps counter (**Line 93**) and print information about the frames per second to our terminal (**Lines 94 and 95**). We close the open window (**Line 98**) followed by stopping the video stream (**Line 99**).

CHAPTER 7

RUNNING SCREENSHOTS OF PROJECT

8.1 REALTIME OBJECT DETECTION

```
D:\realtme Od>python real_time_object_detection.py --prototxt MobileNetSSD_deploy.prototxt.txt --model MobileNetSSD_deploy.caffemodel  
[INFO] loading model...  
[INFO] starting video stream...
```

Fig 8.1.1 Loading model and initializing video stream

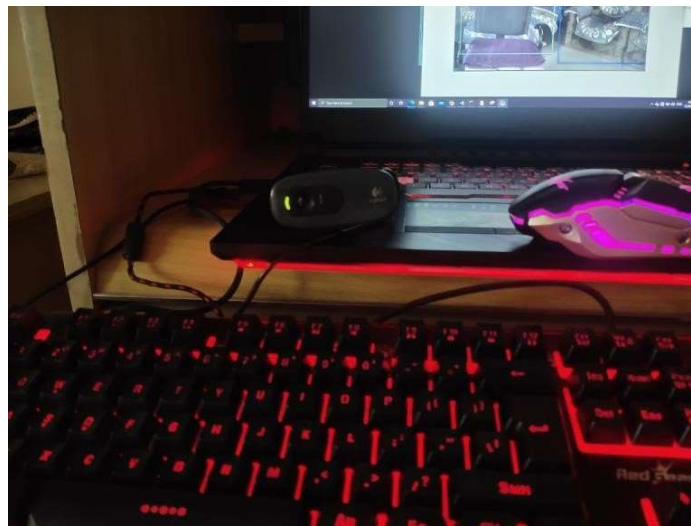


Fig 8.1.2 Initializing Webcam



Fig 8.1.3 Real-time object detection result

CHAPTER 8

APPLICATION

9.1. Application of Real-time Object Detection.

The major applications of Object Detection: -

9.1.1 Facial Recognition. “Deep Face” is a deep learning facial recognition system developed to identify human faces in a digital image. Designed and developed by a group of researchers in Facebook. Google also has its own facial recognition system in Google Photos, which automatically separates all the photos according to the person in the image. There are various components involved in Facial Recognition or authors could say it focuses on various aspects like the eyes, nose, mouth and the eyebrows for recognizing a faces.

9.1.2 People Counting. People counting is also a part of object detection which can be used for various purposes like finding person or a criminal; it is used for analysing store performance or statistics of crowd during festivals. This process is considered a difficult one as people move out of the frame quickly.

9.1.3 Industrial Quality Check. Object detection also plays an important role in industrial processes to identify or recognize products. Finding a particular object through visual examination could be a basic task that's involved in multiple industrial processes like sorting, inventory management, machining, quality management, packaging and so on.

Inventory management can be terribly tough as things are hard to trace in real time. Automatic object counting and localization permits improving inventory accuracy.

9.1.4 Self Driving. Cars Self-driving is the future most promising technology to be used, but the working behind can be very complex as it combines a variety of techniques to perceive their surroundings, including radar, laser light, GPS, odometer, and computer vision. Advanced control systems interpret sensory info to allow navigation methods to work, as well as obstacles and it. This is a big step towards Driverless cars as it happens at very fast speed.

9.1.5 Security Object Detection. plays a vital role in the field of Security; it takes part in major fields such as face ID of Apple or the retina scan used in all the sci-fi movies. Government also widely use this application to access the security feed and match it with their existing database to find any criminals or to detecting objects like car number involved in criminal activities. The applications are limitless

CONCLUSION

Deep learning-based object detection has been a research hotspot in recent years. This project starts on generic object detection pipelines which provide base architectures for other related tasks. With the help of this the three other common tasks, namely object detection, face detection and pedestrian detection, can be accomplished [1]. Authors accomplished this by combining two things: Object detection with deep learning and OpenCV and Efficient, threaded video streams with OpenCV. The camera sensor noise and lightening condition can change the result as it can create problem in recognizing the object. The end result is a deep learning-based object detector that can process around 20-40 FPS. Handwritten mathematical expression recognition of simple algebraic expression process performed successfully. The work included several phases which include data collection, preprocessing, segmentation, feature extraction, classification and symbol recognition. The accuracy of symbol classification and whole algebraic expression recognition is tested. An user interface to automatic mathematical expression recognition is developed with effective classifier. Hence, it is observed that classifier produces higher accuracy. In future, the work can be extended to recognize on-line mathematical expression recognition.

A C K N O W L E D G E M E N T

I feel great pleasure in submitting this Mini-Project report on “**Real Time Object Detection.**” I would wish to thank to our Principal **Dr. R. B. Barjibhe** and **H.O.D., Dr. D. D. Patil** for opening the doors of knowledge towards the realization of this Mini project.

I wish to express true sense of gratitude towards our teacher and guide, **Dr. D. D. Patil** who at every discrete step in study of this Mini-Project contributed his valuable guidance and help me to solve every problem that arose.

Most likely I would like to express my sincere gratitude towards my family and friends for always being there when I needed them the most.

With all respect and gratitude, I would like to thank all authors listed and not listed in references whose concepts are studied and used by me whenever required. I owe my all success to them.

Mr. Nitish Kailas Patil

PRN Number: 2151701242003

T.Y. B. Tech- 2023-2024, SSGBCOET, BSL

REFERENCES:

- [1] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchy for accurate object detection and semantic segmentation. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.
- [2] Ross Girshick. Fast R-CNN. In International Conference on Computer Vision (ICCV), 2015.
- [3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Realtime object detection with region proposal networks. In Advances in Neural Information Processing Systems (NIPS), 2015.
- [4] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
- [5] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, ChengYang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In ECCV, 2016.
- [6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.
- [7] Qi Xiangwei, Xinjiang, “The study of mathematical expression recognition and the embedded system design”, Journal of Software, Vol. 5, No.1, January 2009.
- [8] Nicolas D. Jimenez and Lan Nguyen “Recognition of Handwritten Mathematical Symbols with PHOG Features,” <http://cs229.stanford.edu>.
- [9] Surendra P. Ramteke, Dhanushri V. Patil, Nilima P. Patil, “Neural Network Approach to Mathematical Expression Recognition system”, International journal of engineering research and technical ISSN-2278-0181 Vol. Issue 10 December 2012.
- [10] Hsin-Chia Fu, Memberand Yeong Yuh Xu Multilinguistic “Handwritten Character Recognition by Bayesian Decision-Based Neural Networks”, 1998
- [11] Padmapriya, S. Karpagavall, “Offline Handwritten Mathematical Expression Recognition”, IJIRCCE, Vol. 4, Issue 1, January 2016.