

OBJECT DETECTION WITH DEEP LEARNING

A Project Work

Submitted in the partial fulfilment for the award of the degree of

**BACHELOR OF ENGINEERING
IN**

**ARTIFICIAL INTELLIGENCE AND
MACHINE LEARNING**

Submitted by:

**VUPPUTURI BHARATH
20BCS6586**

Under the Supervision of:

Ms. SUKHMANN KAUR



**CHANDIGARH
UNIVERSITY**
Discover. Learn. Empower.

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
APEX INSTITUTE OF TECHNOLOGY**

**CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413,
PUNJAB**

MAY, 2022

DECLARATION

I, **‘Vupputuri Bharath’**, student of **‘Bachelor of Engineering in Artificial Intelligence and Machine Learning’**, session: **2020-2024**, Department of Computer Science and Engineering, Apex Institute of Technology, Chandigarh University, Punjab, hereby declare that the work presented in this Project Work entitled **‘Object Detection with Deep Learning’** is the outcome of our own bona fide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Date: 11-11-2022

Place: GHOURAN, PUNJAB

(VUPPUTURI BHARATH)

UID: 20BCS6586

TABLE OF CONTENTS

Title Page	1
Declaration of the Student	2
Abstract	5
Acknowledgement	6
Timeline / Gantt Chart	7
CHAPTER1:INTRODUCTION	8 - 14
1.1 Project Objective	
1.2 Motivation	
1.3 Hardware Specifications	
1.4 Software Specifications	
CHAPTER2: LITERATURE SURVEY	15 - 21
CHAPTER 3: PROBLEM FORMULATION	22 - 23
CHAPTER 4: METHODOLOGY	24 - 70
4.1:OBJECT DETECTION	
4.1.1 Introduction to Object Detection	
4.1.2 Digital ImageProcessing	
4.1.3 Gray Scale Image	
4.1.4 ColorImage	
4.1.5 RelatedTechnology	
4.1.6 ApplicationsofObjectDetection	
4.1.7 Objectdetectionworkflowandfeatureextraction	
4.2:DEEPLARNING	
4.2.1 Introduction	
4.2.2 Feed forward and feedback networks	
4.2.3 Weightedsum	
4.2.4 Activationfunction	

4.3:CONVOLUTIONNEURALNETWORKS

- 4.3.1 Introduction
- 4.3.2 Convolutional Neural Networks
- 4.3.3 CNNArchitecture
- 4.3.4 OverallArchitecture
- 4.3.5 ConvolutionalLayers

4.4:OPENCOMPUTERVISION

- 4.4.1 Introduction
- 4.4.2 Applications
- 4.4.3 LibrariesinOpenCV
- 4.4.4 HaarCascadeClassfiers

CHAPTER 5: RESULTS AND DISCUSSIONS 71 - 74

6. IMPLEMENTATION SNAPSHOTS OF SOURCE CODE 75 - 84

7. CONCLUSIONS 85

8. FUTURE ENHANCEMENTS 86 - 89

9. REFERENCES 90

ABSTRACT

Real time object detection may be a huge, spirited and sophisticated space of computer vision. If there may be a single object to be detected in a picture, it's called Image Localization and if there are multiple objects in a picture, then it's Object Detection. This detects the semantic objects of a category in digital images and videos. The applications of real time object detection include tracking objects, video surveillance, pedestrian detection, folks count, self-driving cars, face detection, ball tracking in sports and plenty of additional. Convolution Neural Networks may be a representative tool of Deep learning to find objects victimisation OpenCV(Opensource computer Vision), that may be a library of programming functions mainly geared toward Realtime computer vision.

Keywords: Computer vision, Deep Learning, Convolution Neural Networks.

ACKNOWLEDGEMENT

It gives us immense pleasure to express our deepest sense of gratitude and sincere thanks to our respected guide Ms. Sukhman kaur (Assistant Professor), CSE- Artificial Intelligence, Chandigarh University, Mohali for his valuable guidance, encouragement and help for completing this work. His useful suggestions for this whole work and co-operative behaviour are sincerely acknowledged. We are also grateful to Dr. Shikha Gupta (Program Leader, CSE-AIML) for her constant support and guidance.

We also wish to express our indebtedness to our family members whose blessings and support always helped us to face the challenges ahead. We also wish to express thanks to all people who helped us in completion of this project.

AMAN SHUKLA (20BCS6568)

AMAN MISHRA (20BCS6594)

KETHURI AJAY (20BCS6585)

VUPPUTURI BHARATH (20BCS6586)

PLACE: GHARUAN, MOHALI

DATE: 11-11-2022

Timeline / Gantt Chart

S.N	Strategies	1 st week	2 nd week	3 rd week	4 th week	5 th week	6 th week
1)	Problem Identification						
2)	Research & Analysis						
3)	Design						
4)	Coding						
5)	Implementation & testing						
6)	Project finalisation						
7)	Documentation						

CHAPTER 1

INTRODUCTION

INTRODUCTION:

Object detection is a widely known research direction in the fields of data science, artificial intelligence, machine learning and computer vision [2]. It is the building block for many more complex projects like object tracking, behavior analysis and facial recognition [2]. It is designed to locate the target object and accurately categories the object. It is an example of a supervised learning algorithm as it requires labeled data as a prerequisite to training and constructing the model. understand the underlying pattern hidden to be able to successfully yield satisfactory results on the testing data. A study performed in 2018 states that the current trends state that the fields of computer vision and artificial intelligence will see a big rise due to many industries and other sectors relying on automation including sectors like medical, defense and also other sectors such as agriculture which are making a swift change towards automation and artificial intelligence [3].

Gartner's 2018 technology trend states that Artificial Intelligence would be widely used trend among the industries and so the Computer vision. Industries based on automation, consumer markets, medical domains, defense and surveillance sectors are most likely domains extensively using computer vision. It is forecasted that CV market would reach \$33.3 billion in 2019 fostering the remarkable growth in the domains of consumer, robotics, and machine vision

As early as 1956, the concept of "artificial intelligence" was proposed. But it was not until 2012 that artificial intelligence began to usher in a peak. This is mainly due to the rising data volume, computing power and the emergence of machine learning algorithms. The development of detection systems is closely related to the explosion of data volume. This is because the performance test and algorithm evaluation need to be obtained through dataset, and dataset is also a powerful driving force to promote the research field of detection approaches.

Deep Learning word has become a famous domain nowadays within computer vision mainly because of the highly successful results produced by deep learning algorithms in the fields of image classification, object detection and natural language processing [3].

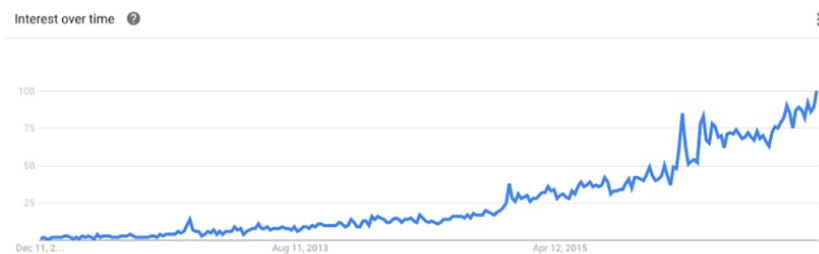


Fig.1 The rising trend of deep learning in the last decade.

Deep Neural architectures handle complex image processing and detection models better than normal shallow algorithms [3]. Deep Neural networks like CNN shows average accuracy for smaller and simpler data but represents very high accuracy on large datasets of images [5]. This shows the rising trend of deep learning used in image processing and objects detection.

In General, Object Detection techniques with deep learning are divided into two categories: direct object prediction in which we directly predict the object in the boundaries without any intermediate task in between, it means in a one stage fashion

[6]. This leads to faster response but the model is not flexible for a change in tasks. Other category is of two stage detection which implements the concept of regression algorithms [6]. Both of the categories are dealt with detail in the later sections.

Image classification, being the widely researched area in the domain of computer vision has achieved remarkable results in world-wide competitions such as ILSVRC, PASCAL VOC, and Microsoft COCO with the help of deep learning. Motivated by the results of image classification, deep learning models have been developed for object detection and deep learning based object detection has also achieved state-of-the-results.

Due to object detection's close relationship with video analysis and image understanding, it has attracted much research attention in recent years. Traditional object detection methods are built on handcrafted features and shallow trainable architectures. Their performance easily stagnates by constructing complex ensembles which combine multiple low-level image features with high-level context from object detectors and scene classifiers. With the rapid development in deep learning, more powerful tools, which are able to learn semantic, high-level, deeper features, are introduced to address the problems existing in traditional architectures. These models behave differently in network architecture, training strategy and optimization function, etc.

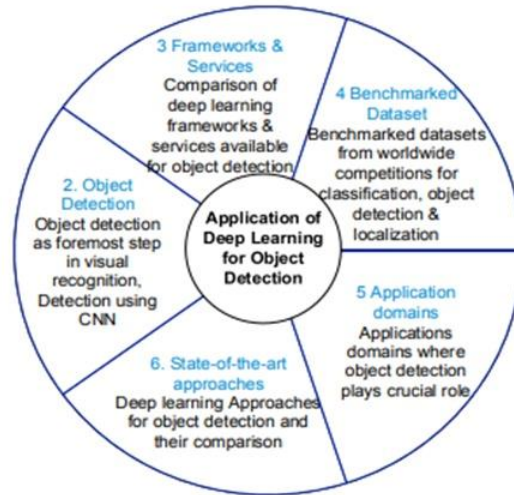


Fig.2 How Deep learning plays a vital role in Object detection and its application [3]

Object detection is a basic research direction in the fields of computer vision, deep learning, artificial intelligence, etc. It is an important prerequisite for more complex computer vision tasks, such as target tracking, event detection, behavior analysis, and scene semantic understanding. It aims to locate the target of interest from the image, accurately determine the category and give the bounding box of each target.

The list of deep learning frameworks available till date is exhaustive. We have mentioned some significant deep learning frameworks in table 2. The frameworks are studied from the point of view of features exhibited, interface, support for deep learning model viz. convolution neural network, recurrent neural network (RNN), Restricted Boltzmann Machine (RBM) and Deep Belief Network (DBN) and support for Multi-node parallel execution, developer of the framework and license

Multi-type detection, posture detection, face detection, and pedestrian detection is in the realm of image object detection research. We will also encounter devices that apply image object detection in our daily life. To date, benchmarks such as Caltech, ImageNet, PASCAL VOC [2], and MS COCO have played an important role in the area of object detection

1.1 Project Objective:

The motive of object detection is to recognize and locate all known objects in a scene. Preferably in 3D space, recovering pose of objects in 3D is very important for robotic control systems.

Imparting intelligence to machines and making robots more and more autonomous and independent has been a sustaining technological dream for the mankind. It is our dream to let the robots take on tedious, boring, or dangerous work so that we can commit our time to more creative tasks. Unfortunately, the intelligent part seems to be still lagging behind. In real life, to achieve this goal, besides hardware development, we need the software that can enable robot the intelligence to do the work and act independently. One of the crucial components regarding this is vision, apart from other types of intelligences such as learning and cognitive thinking. A robot cannot be too intelligent if it cannot see and adapt to a dynamic environment.

The searching or recognition process in real time scenario is very difficult. So far, no effective solution has been found for this problem. Despite a lot of research in this area, the methods developed so far are not efficient, require long training time, are not suitable for real time application, and are not scalable to large number of classes. Object detection is relatively simpler if the machine is looking for detecting one particular object. However, recognizing all the objects inherently requires the skill to differentiate one object from the other, though they may be of same type. Such problem is very difficult for machines, if they do not know about the various possibilities of objects.

1.2 Motivation:

Blind people do lead a normal life with their own style of doing things. But, they definitely face troubles due to inaccessible infrastructure and social challenges. The biggest challenge for a blind person, especially the one with the complete loss of vision, is to navigate around places. Obviously, blind people roam easily around their

house without any help because they know the position of everything in the house. Blind people have a tough time finding objects around them. So we decided to make a REAL TIME OBJECT DETECTION System. We are interested in this project after we went through few papers in this area. As a result we are highly motivated to develop a system that recognizes objects in the real time environment.

1.3 Hardware Specifications

- Processor (CPU) with 2.8 gigahertz (GHz) frequency or above.
- A minimum of 4 GB of RAM.
- Monitor Resolution 1024 X 768 or higher.
- A minimum of 20 GB of available space on the hard disk.
- Internet Connection Broadband (high-speed) Internet connection with a speed of 3 Mbps.
- Keyboard and a Mouse or some other compatible pointing device.

1.4 Software Specifications

PYTHON:

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming.

JUPYTER-ANACONDA:

The Jupyter Notebook application allows you to create and edit documents that display the input and output of a Python or R language script. Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012. As an Anaconda, Inc. product, it is also known as Anaconda Distribution or Anaconda Individual Edition, while other products from the company are Anaconda Team Edition and Anaconda Enterprise Edition, both of which are not free.

CHAPTER 2

LITERATURE SURVEY

LITERATURE REVIEW:

There are a lot of innovative and inspiring ideas for object detection models which are constructed either from a business standpoint or research programs. Some of which will be dealt with detail in this module:

S. Manjula [1] proposed a model to replace all the human labored detection systems with machine learned models. This model implies the use of background modeling to extract the object from the background and motion segmentation to track independent moving objects [7]. The goal of object detection is to detect all instances of objects from a known class, such as people, vehicles or faces in an image or video. Object detection is a difficult task because of illumination changes in environment, rapid variations in target appearance, similar non-target objects in background, and occlusions. Object detection method uses semiautomatic or automatic detection techniques. Moreover it makes the use of object classification to distinguish objects and behavioural analysis to predict high level description of actions.

Sl.No	Method	Accuracy	Computational time
1	Shape-based classification	Moderate	Low
2	Motion-based classification	Moderate	High
3	Texture-based classification	High	High

Fig.4 Table of comparison for object classification methods from the paper.

The model was successful in fulfilling the task of building a automatic detection system which can be used in place of a human powered one but the methods compared in the models were either low accuracy and low computational power or having significant accuracy but requires high computational power which can disrupt the cost side of the system from a financial point.

Jun Deng et al [2] in 2020 compared two types of target detection frameworks namely one stage and two stage detections. The Algorithms used for one stage detections were various modifications of YOLO (You only look once) which are faster and provides high performances as it make use of one stage detection. In comparison to the models used by two stage detection models like R-CNN [8] or faster R-CNN [9] and its variations have reached record breaking accuracy but are very high power consuming algorithms. Models like SSD [10] proposed by Liu were able to find a middle path with moderate speeds with less power consuming algorithms.

TABLE II. COMPARISON OF OBJECT DETECTION ALGORITHMS

Method	Backbone	Size/Pixel	Test	mAP/%	fps
YOLOv1	VGG16	448×448	VOC 2007	66.4	45
SSD	VGG16	300×300	VOC 2007	77.2	46
YOLOv2	Darknet-19	544×544	VOC 2007	78.6	40
YOLOv3	Darknet-53	608×608	MS COCO	33	51
YOLOv4	CSP Darknet-53	608×608	MS COCO	43.5	65.7
R-CNN	VGG16	1000×600	VOC2007	66	0.5
SPP-Net	ZF-5	1000×600	VOC2007	54.2	-
Fast R-CNN	VGG16	1000×600	VOC2007	70.0	7
Faster R-CNN	ResNet-101	1000×600	VOC2007	76.4	5

Fig.5 Comparison table for object detection model from the paper.

The paper concluded that future research should reduce the dependence of data and to achieve efficient detection of small objects. Further, As one of the most basic and challenging problems in computer vision, object detection has received great attention

in recent years. Detection algorithms based on deep learning have been widely applied in many fields, but deep learning still has some problems to be explored:

- 1) Reduce the dependence on data.
- 2) To achieve efficient detection of small objects.
- 3) Realization of multi-category object detection

Ajeet Ram Pathak [3] introduced the frameworks and services used in research for object detection. Datasets that are reviewed and benchmarked by companies like Microsoft and Caltech are available to improve the current object detection models. Moreover Deep Learning Frameworks and services available for object detection with their respective features, Interface, softwares and which deep learning models it accompanies and licensing required are available in a tabular format to help new developers in the particular field to learn the basics of the object detection domain. In this paper, they demystified the role of deep learning techniques based on CNN for object detection. Deep learning frameworks and services available for object detection are also discussed in the paper. Benchmarked datasets for object localization and detection released in worldwide competitions are also covered. The pointers to the domains in which object detection is applicable has been discussed. State-of-the-art deep learning based object detection techniques have been assessed and compared.

Method	Working	Features
Deep saliency network	CNNs are used for extracting the high-level and multi-scale features.	It is challenging to detect the boundaries of salient regions due to the fact that pixel residing in the boundary region have similar receptive fields. Due to this, network may come with inaccurate map and shape of the object to be detected.
Generating image (or pixels)	This method is used when the occurrence of occlusions and deformations is rare in the dataset.	This method generates new images with occlusions and deformations only when training data contains occurrences of occlusions and deformations.
Generating all possible occlusions and deformations	In this method, all sets of possible occlusions and deformations are generated to train the object detectors.	This method is not scalable since deformations and occlusions incur large space.
Adversarial learning	Instead of generating all deformations and occlusions, this method use adversarial network which selectively generates features mimicking the occlusions and deformations which are hard to be recognized by the object detector.	As this method generates the examples on-the fly, it is good candidate to be applied in real time object detection. As it selectively generates the features, it is also scalable.
Part-based method	This method represents object as collection of local parts and spatial structure. This method exhaustively searches for multiple parts for object detection.	This method addresses the issue of intra-class variations in object categories Such variations occur due to variation in poses, cluttered background, partial occlusions
CNN with part-based method	In this method, deformable part model is used for modelling the spatial structure of the local parts whereas CNN is used for learning the discriminative features.	This method handles the issue of partial occlusions. But requires multiple CNN models for part based object detection. Finding out the optimal number of parts per object is also challenging.
Fine-grained object detection method	This methods works on annotated object parts during training phase. Part-localization is the fundamental component used in testing phase.	This method has capability to figure out the differences in inter-class objects at finer level. And they work more on discriminative parts compared to generic object detection methods.

Fig.6 Table of Comparision of deep learning based detection models discussed in paper.

Zhong-Qiu Zhao [5] introduced with the brief overview and history and prosperity of deep learning. Generic object detection different from general object detection aims to recognize existing objects fed to the machine learning models with boxes around the object. Salient object detection is the most challenging field of object detection is used to find the most dominant objects present in a image. Various applications accompany salient object detection such as image cropping, image retrieval and even in investigations to highlight clues. Facial Detection is essentially the more complex form of object detection but captures human faces and is basically the prerequisites for facial recognition, facial synthesis and facial expression analysis which are many different domains inside the image processing field.

Method	Reasonable	All	Far	Medium	Near	none	partial	heavy
Checkerboards+ [198]	17.1	68.4	100	58.3	5.1	15.6	31.4	78.4
LDCF++[S2]	15.2	67.1	100	58.4	5.4	13.3	33.3	76.2
SCF+AlexNet [210]	23.3	70.3	100	62.3	10.2	20.0	48.5	74.7
SA-FastRCNN [211]	9.7	62.6	100	51.8	0	7.7	24.8	64.3
MS-CNN [105]	10.0	61.0	97.2	49.1	2.6	8.2	19.2	60.0
DeepParts [204]	11.9	64.8	100	56.4	4.8	10.6	19.9	60.4
CompACT-Deep [195]	11.8	64.4	100	53.2	4.0	9.6	25.1	65.8
RPN+BF [203]	9.6	64.7	100	53.9	2.3	7.7	24.2	74.2
F-DNN+SS [207]	8.2	50.3	77.5	33.2	2.8	6.7	15.1	53.4

Fig.7 Performance comparisons for state of the art models introduced in the paper.

This paper provides a detailed review on deep learning based object detection frameworks which handle different sub-problems, such as occlusion, clutter and low resolution, with different degrees of modifications on R-CNN. The review starts on generic object detection pipelines which provide base architectures for other related tasks. Then, three other common tasks, namely salient object detection, face detection and pedestrian detection, are also briefly reviewed. Finally, they propose several promising future directions to gain a thorough understanding of the object detection landscape. This review is also meaningful for the developments in neural networks and related learning systems, which provides valuable insights and guidelines for future progress.

Jun Wang [6] proposed a method for bounding the spatial location and space around the target variable inside an image or a video with the help of background subtraction, object localization and semantic segmentation. Moreover the structure and working of many known object detection algorithms are discussed and defined. Evaluation metrics such as confusion matrix and precision, recall are detailed. Furthermore, the Development and evolution of object detection technologies are discussed. Various application of object detection can be pedestrian detection to reduce accidents, facial and text detection and remote sensing object detection which is still a research direction within object detection.

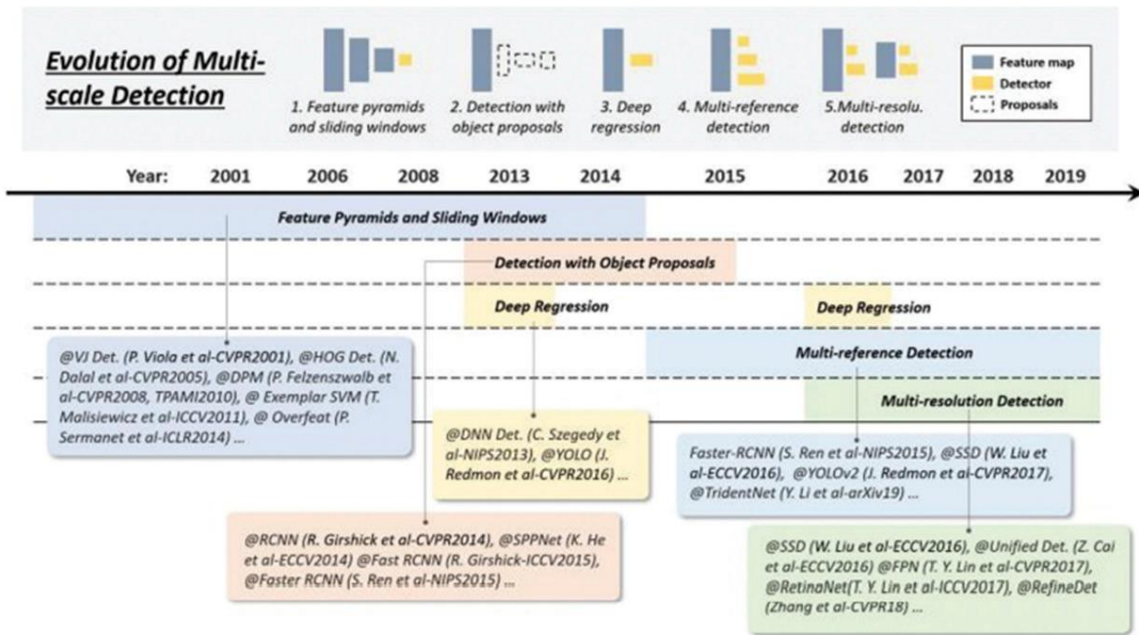


Fig.8 Evolution of detection models over the years cited in the paper.

LITERATURE REVIEW SUMMARY:

Year and citation	Article Title	Purpose of study	Tools / Software Used	Comparison of technique	Source (Journal / Conference)	Findings	Data set (if used)	Evaluation parameters
2016	A Study On Object Detection	Research Article						
2020	A review of research on object detection based on deep learning	Research Article					VOOC 2007, MS COCO	
2018	Application of Deep Learning for Object Detection	Research Article					ILSVRC	
2019	Object Detection with Deep Learning: A Review	Research Article					VOC, COCO	
2021	Deep Learning for Object Detection: A Survey	Research Article					Image Net, DOTA, VOC, COCO	

CHAPTER 3

PROBLEM FORMULATION

PROBLEM FORMULATION

During project development, problems can occur in software intentionally or unintentionally. Developers tend to create copies of projects so that the solution of a problem can be resolved in multiple ways and the best outcome can be taken out for further work.

From the literature review, it is observed that several experiments are done on MS COCO datasets to demonstrate that RefinedD achieves state-of-the-art detection accuracy with high efficiency. Also, several different machine models have been taken up for detection of objects either in live feeds or from images.[1]

With this, the problems can be resolved by giving the better weights to model after repeating the steps of training. So, problems may arise in that. But it can be formulated and resolved by presenting correct conditions for different scenarios.

CHAPTER 4

METHODOLOGY

METHODOLOGY:

The following methodology will be followed to achieve the objectives defined for proposed research work:

1. Our object detection application work on the principle of opencv which resides on the framework of deep learning.
2. The pre-trained model used for the task is ssd-mobile net which is trained upon coco dataset which contains a total of 91 image classes and labels that the model can successfully detect with appropriate accuracy.
3. Our project works with three function one uses images as input the other option take videos as images and detect objects in the video with the timestamp and finally one which uses the webcam and take live video from webcam as input and detect objects in front of the system.
4. The graphical user interface used for the frontend is made with tkinter library and successfully transforms ordinary hard to navigate independent functions to an interactive easy to navigate and operate complete application with all the functionality.

IN DEPTH ANALYSIS OF PROJECT:-

1. For the training data we provide the model with plenty of training images and class labels of the objects present in the images so the model can learn which object has to be classified in which label. Many open source datasets for object detection containing the class labels are provided on the web to make the task easier.
2. Next is to load the model into the system many pre trained models are present online and they work well as they tend generalize well. One such model is ssd mobile net which works very well for object detection and provide faster results.
3. Since the Opencv works on the principle of data flow graph we have to download the frozen inference graph for our particular model that is supported from the web. The frozen inference graph is needed to be loaded into the memory.
4. Now to work with Opencv we have to convert the images into the numpy array. So we use the reshape feature of the numpy libraries to convert the complex images into arrays so it is easier to process for the Opencv.
5. Next, is to provide the test input in the model to check whether our model is working well or not. We can choose any number of images and store it in a folder whose path is to be given.
6. The output containing detection boxes means the boxes in which the object is enclosed is formed along with detection classes and detection scores which indicate the likelihood of matching the predicted result with actual object.

4.1:OBJECT DETECTION

4.1.1 Introduction to Object Detection

Object Detection is the process of finding and recognizing real-world object instances such as car, bike,TV,flowers,andhumansoutofanimagesorvideos.Anobjectdetectiontechniquelets youunderstandthe details of an image or a video as it allows for the recognition, localization, and detection of multipleobjectswithinanimage.

Itisusually utilized inapplicationslikeimageretrieval, security,surveillance,and advanceddriverassistancesystems (ADAS).ObjectDetectionisdonethrough manyways:

- FeatureBasedObjectDetection
- ViolaJonesObjectDetection
- SVMClassificationswithHOGFeatures
- DeepLearningObjectDetection

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

The detection ofanobject in video sequenceplays amajor rolein severalapplications specificallyasvideosurveillanceapplications.

Objectdetectioninavideostreamcanbedonebyprocesseslikepre-processing,segmentation,foregroundandbackgroundextraction, featureextraction.

Humans can easily detect and identify objects present in an image. The human

visual system is fast and accurate and can perform complex tasks like identifying multiple objects with little conscious thought. With the availability of large amounts of data, faster GPUs, and better algorithms, we can now easily train computers to detect and classify multiple objects within an image with high accuracy.

4.1.2 Digital Image Processing

Computerized picture preparing is a range portrayed by the requirement for broad test work to build up the practicality of proposed answers for a given issue. A critical trademark hidden the plan of picture preparing frameworks is the huge level of testing and experimentation that

Typically is required before touching base at a satisfactory arrangement. This trademark informs that the capacity to plan approaches and rapidly model hopeful arrangements by and large assumes a noteworthy part in diminishing the cost and time required to land a suitable framework execution.

WHAT IS DIP?

A picture might be characterized as a two-dimensional capacity $f(x, y)$, where x, y are spatial directions, and the adequacy of a function of directions (x, y) is known as the power or dark level of the picture by then. Whenever x, y and the abundance estimation of a real limited discrete amounts, we call the picture a computerized picture. The field of DIP alludes to preparing advanced picture by methods for computerized PC. Advanced picture is made out of a limited number of components, each of which has a specific area and esteem. The components are called pixels.

Vision is the most progressive of our sensor, so it is not amazing that picture play the absolute most imperative part in human observation. Be that as it may, dissimilar to people, who are constrained to the visual band of the EM range imaging machines cover practically the whole EM range, going from gamma to radio waves. They can work likewise on pictures produced by sources that people are not acclimated to partner with picture.

There is no broad understanding among creators in regards to where picture handling stops and other related territories, for example, picture examination and PC vision begin. Now and then a qualification is made by characterizing picture handling as a teach in which both the info and yield at a procedure are pictures. This is constraining and to some degree manufactured limit. The range of picture investigation is in the middle of picture preparing and PC vision.

There are no obvious limits in the continuum from picture handling toward one side to finish vision at the other. In any case, one helpful worldview is to consider three sorts of mechanized procedures in this continuum: low, mid and abnormal state forms. Low-level process includes primitive operations, for example, picture preparing to decrease commotion differentiate upgrade and picture honing. A low-level process is described by the way that both its sources of info and yields are pictures.

Mid-

level process on pictures includes assignments, for example, division, depiction of that Question diminish them to a frame reasonable for PC handling and characterization of individual articles

A mid-level process is portrayed by the way that its sources of info by and large are pictures however its yields are properties removed from those pictures. At long last more clavated amount handling includes "Understanding an outlet of perceived items, as in picture examination and at the farthest end of the continuum playing out

the intellectual capacities typically connected with human vision. Advanced picture handling, as effectively characterized is utilized effectively in a wide scope of regions of outstanding social and monetary esteem.

WHAT IS AN IMAGE?

A picture is spoken to as a two dimensional capacity $f(x, y)$ where x and y are spatial co-ordinates and the adequacy of "T" at any match of directions (x, y) is known as the power of the picture by then.



Fig. 1.1 digital image

Processing on image:

Processing on image can be of three types They are low-level, mid-level, high level.

Low-level Processing:

- Preprocessing to remove noise.
- Contrast enhancement.
- Image sharpening.

Medium Level Processing :

- Segmentation.
- Edge detection
- Object extraction.

High Level Processing:

- Image analysis
- Scene interpretation

Why Image Processing?

Since the digital image is invisible, it must be prepared for viewing on one or more output device(laser printer, monitor at).The digital image can be optimized for the application by enhancing the appearance of the structures within it.

There are three of image processing used. They are

- Image to Image transformation
- Image to Information transformations
- Information to Image transformations

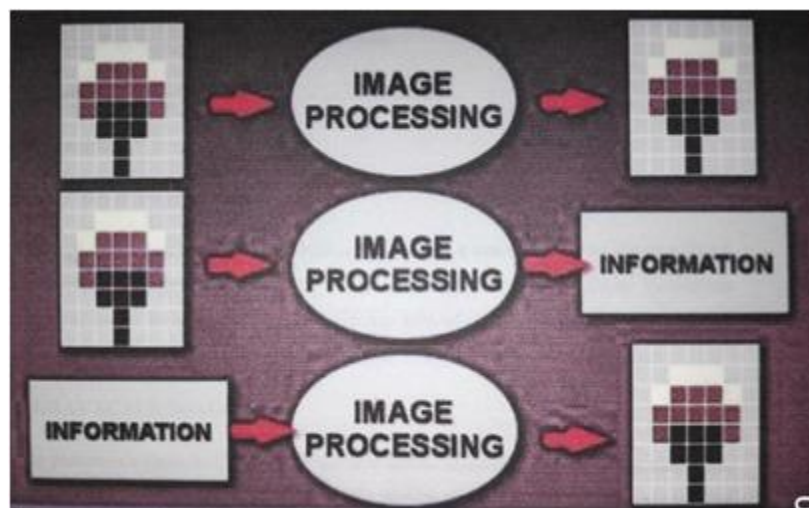


Fig. 1.2 Types of Image Processing

Pixel :

Pixel is the smallest element of an image. Each pixel correspond to any one value. In an 8-bit gray scale image, the value of the pixel between 0 and 255.Each pixel store a value proportional to the light intensity at that particular location. It is indicated in either Pixels per inch or Dots per inch.

Resolution:

The resolution can be defined in many ways. Such as pixel resolution, spatial resolution, temporal resolution, spectral resolution. In pixel resolution, the term resolution refers to the total number of count of pixels in a digital image. For example, If an image has M rows and N columns, then its resolution can be defined as $M \times N$. Higher is the pixel resolution, the higher is the quality of the image.

Resolution of an image is of generally two types.

- Low Resolution image
- High Resolution

Since high resolution is not a cost effective process It is not always possible to achieve high resolution images with low cost. Hence it is desirable Imaging. In Super Resolution imaging, with the help of certain methods and algorithms we can be able to produce high resolution images from the low-resolution image from the low resolution images.

4.1.3 Gray Scale Image

A gray scale picture is a capacity $I(x, y)$ of the two spatial directions of the picture plane. $I(x, y)$ is the force of the picture force of picture at the point (x, y) on the picture plane. $I(x, y)$ take non- negative expect the picture is limited by a rectangle

4.1.4 ColorImage

It can be spoken to by three capacities, $R(x, y)$ for red, $G(x, y)$ for green and $B(x, y)$ for blue. A picture might be persistent as for the x and y facilitates and furthermore in adequacy. Changing oversuch a picture to advanced shape requires that the directions and the adequacy to be digitized. Digitizing the facilitate's esteems is called inspecting. Digitizing the adequacy esteems is called quantization.

4.1.5 Related Technology

R-CNN

R-CNN is a progressive visual object detection system that combines bottom-up region proposals with rich options computed by a convolution neural network.

R-CNN uses region proposal ways to initial generate potential bounding boxes in a picture and then run a classifier on these proposed boxes.

SINGLE SIZE MULTI BOX DETECTOR

SSD discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At the time of prediction the network generates scores for the presence of each object category in each default box and generates adjustments to the box to better match the object shape.

Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes.

ALEXNET

AlexNet is a convolutional neural Network used for classification which has 5 Convolutional layers, 3 fullyconnected layers and 1 softmax layer with 1000 outputs for classification as his architecture.

YOLO

YOLO is real-time object detection. It applies one neural network to the complete image dividing the image into regions and predicts bounding boxes and possibilities for every region.

Predicted probabilities are the basis on which these bounding boxes are weighted. A single neural network predicts bounding boxes and class possibilities directly from full pictures in one evaluation. Since the full detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

VGG

VGG network is another convolution neural network architecture used for image classification.

MOBILENETS

To build lightweight deep neural networks MobileNets are used. It is based on a streamlined architecture that uses depth-wise separable convolutions. MobileNet uses 3×3 depth-wise separable convolutions that uses between 8 times less computation than standard convolution at solely a little reduction accuracy. Applications and use cases including object detection, fine grain classification, face attributes and large scale-localization.

TENSOR FLOW

Tensor flow is an open source software library for high performance numerical computation. It allows simple deployment of computation across a range of platforms (CPUs, GPUs, TPUs) due to its versatile design also from desktops to clusters of servers to mobile and edge devices. Tensor flow was designed and developed by researchers and engineers from the Google Brain team at intervals Google's AI organization, it comes with robust support for machine learning and deep learning and the versatile numerical computation core is used across several alternative scientific domains.

To construct, train and deploy Object Detection Models TensorFlow is used that makes it easy and also it provides a collection of Detection Models pre-trained on the COCO dataset, the Kitti dataset, and the Open Images dataset. One among the numerous Detection Models is that the combination of Single Shot Detector (SSDs) and Mobile Nets architecture that is quick, efficient and doesn't need huge computational capability to accomplish the object Detection.

4.1.6 Applications of Object Detection

The major applications of Object Detection are:

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.

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.

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.

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.

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.

4.1.7 Object detection workflow and feature extraction

Every Object Detection Algorithm works on the same principle and it's just the working that differs from others. They focus on extracting features from the images that are given as the input at hands and then it uses these features to determine the class of the image.

4.2:DEEP LEARNING

4.2.1 Introduction

Deep learning is a machine learning technique. It teaches a computer to filter inputs through layers to learn how to predict and classify information. Observations can be in the form of images, text, or sound. The inspiration for deep learning is the way that the human brain filters information. Its purpose is to mimic how the human brain works to create some real magic. In the human brain, there are about 100 billion neurons. Each neuron connects to about 100,000 of its neighbors. We're kind of recreating that, but in a way and at a level that works for machines. In our brains, a neuron has a body, dendrites, and an axon. The signal from one neuron travels down the axon and transfers to the dendrites of the next neuron. That connection where the signal passes is called a synapse. Neurons by themselves are kind of useless. But when you have lots of them, they work together to create some serious magic. That's the idea behind a deep learning algorithm! You get input from observation and you put your input into one layer. That layer creates an output which in turn becomes the input for the next layer, and so on. This happens over and over until your final output signal! The neuron (node) gets a signal or signals (input values), which pass through the neuron. That neuron delivers the output signal.

Think of the input layer as your senses: the things you see, smell, and feel, for example. These are independent variables for one single observation. This information is broken down into numbers and the bits of binary data that a computer can use. You'll need to either standardize or normalize these variables so that they're within the same range. They use many layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output of the previous layer for its input. What they learn forms a hierarchy of concepts. In this hierarchy, each level learns to transform its input data into a more and more abstract and composite

representation. That means that for an image, for example, the input might be a matrix of pixels. The first layer might encode the edges and compose the pixels. The next layer might compose an arrangement of edges. The next layer might encode a nose and eyes. The next layer might recognize that the image contains a face, and so on.

What happens inside the neuron?

The input node takes in information in a numerical form. The information is presented as an activation value where each node is given a number. The higher the number, the greater the activation. Based on the connection strength (weights) and transfer function, the activation value passes to the next node. Each of the nodes sums the activation values that it receives (it calculates the weighted sum) and modifies that sum based on its transfer function. Next, it applies an activation function. An activation function is a function that's applied to this particular neuron. From that, the neuron understands if it needs to pass along a signal or not.

Each of the synapses gets assigned weights, which are crucial to Artificial Neural Networks (ANNs). Weights are how ANNs learn. By adjusting the weights, the ANN decides to what extent signals get passed along. When you're training your network, you're deciding how the weights are adjusted.

The activation runs through the network until it reaches the output nodes. The output nodes then give us the information in a way that we can understand. Your network will use a cost function to compare the output and the actual expected output. The model performance is evaluated by the cost function. It's expressed as the difference between the actual value and the predicted value.

There are many different cost functions you can use, you're looking at what the error you have in your network is. You're working to minimize loss function. (In essence, the lower the loss function, the closer it is to your desired output). The information goes back, and the neural network begins to learn with the goal of minimizing the cost function by tweaking the weights. This process is called backpropagation.

In forward propagation, information is entered into the input layer and propagates forward through the network to get our output values. We compare the values to our expected results. Next, we calculate the errors and propagate the info backward. This allows us to train the network and update the weights. (Backpropagation allows us to adjust all the weights simultaneously.) During this process, because of the way the algorithm is structured, you're able to adjust all of the weights simultaneously. This allows you to see which part of the error each of your weights in the neural network is responsible for.

When you've adjusted the weights to the optimal level, you're ready to proceed to the testing phase!

How does an artificial neural network learn?

There are two different approaches to get a program to do what you want. First, there's the specifically guided and hard-programmed approach. You tell the program exactly what you want it to do. Then there are neural networks. In neural networks, you tell your network the inputs and what you want for the outputs, and then you let it learn on its own.

By allowing the network to learn on its own, you can avoid the necessity of entering in all of the rules. You can create the architecture and then let it go and learn. Once it's trained up, you can give it a new image and it will be able to distinguish output.

4.2.2 Feed forward and feedback networks

A feed forward network is a network that contains inputs, outputs, and hidden layers. The signals can only travel in one direction (forward). Input data passes into a layer where calculations are performed. Each processing element computes based upon the weighted sum of its inputs. The new values become the new input values that feed the next layer (feed-forward). This continues through all the layers and determines the output. Feed forward networks are often used in, for example, data mining.

A feedback network (for example, a recurrent neural network) has feedback paths. This means that they can have signals traveling in both directions using loops. All possible connections between neurons are allowed. Since loops are present in this type of network, it becomes a non-linear dynamic system which changes continuously until it reaches a state of equilibrium. Feedback networks are often used in optimization problems where the network looks for the best arrangement of interconnected factors.

4.2.3 Weighted sum

Inputs to a neuron can either be features from a training set or outputs from the neurons of a previous layer. Each connection between two neurons has a unique synapse with a unique weight attached. If you want to get from one neuron to the next, you have to travel along the synapse and pay the “toll” (weight). The neuron then applies an activation function to the sum of the weighted inputs from each incoming synapse. It passes the result on to all the neurons in the next layer. When we talk about updating weights in a network, we’re talking about adjusting the weights on these synapses.

A neuron’s input is the sum of weighted outputs from all the neurons in the previous layer. Each input is multiplied by the weight associated with the synapse connecting the input to the current neuron. If there are 3 inputs or neurons in the previous layer, each neuron in the current layer will have 3 distinct weights: one for each synapse.

In a nutshell, the activation function of a node defines the output of that node.

The activation function (or transfer function) translates the input signals to output signals. It maps the output values on a range like 0 to 1 or -1 to 1. It's an abstraction that represents the rate of action potential firing in the cell. It's a number that represents the likelihood that the cell will fire. At it's simplest, the function is binary: yes (the neuron fires) or no (the neuron doesn't fire). The output can be either 0 or 1 (on/off or yes/no), or it can be anywhere in a range. If you were using a function that maps a range between

0 and 1 to determine the likelihood that an image is a cat, for example, an output of 0.9 would show a 90% probability that your image is, in fact, a cat.

4.2.4 Activation function

In a nutshell, the activation function of a node defines the output of that node.

The activation function (or transfer function) translates the input signals to output signals. It maps the output values on a range like 0 to 1 or -1 to 1. It's an abstraction that represents the rate of action potential firing in the cell. It's a number that represents the likelihood that the cell will fire. At it's simplest, the function is binary: yes (the neuron fires) or no (the neuron doesn't fire). The output can be either 0 or 1 (on/off or yes/no), or it can be anywhere in a range.

What options do we have? There are many activation functions, but these are the four very common ones:

Thresholdfunction

This is a step function. If the summed value of the input reaches a certain threshold the function passes on 0. If it's equal to or more than zero, then it would pass on 1. It's a

very rigid, straightforward, yes or no function.

Sigmoid function

This function is used in logistic regression. Unlike the threshold function, it's a smooth, gradual progression from 0 to 1. It's useful in the output layer and is used heavily for linear regression.

Hyperbolic Tangent Function

This function is very similar to the sigmoid function. But unlike the sigmoid function which goes from 0 to 1, the value goes below zero, from -1 to 1. Even though this isn't a lot like what happens in a brain, this function gives better results when it comes to training neural networks. Neural networks sometimes get "stuck" during training with the sigmoid function. This happens when there's a lot of strongly negative input that keeps the output near zero, which messes with the learning process.

Rectifier function

This might be the most popular activation function in the universe of neural networks. It's the most efficient and biologically plausible. Even though it has a kink, it's smooth and gradual after the kink at

0. This means, for example, that your output would be either "no" or a percentage of "yes." This function doesn't require normalization or other complicated calculations.

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.

4.3: CONVOLUTION NEURAL NETWORKS

4.3.1 Introduction

Artificial Neural Networks

The idea of ANNs is based on the belief that working of human brain by making the right connections, can be imitated using silicon and wires as living neurons and dendrites.

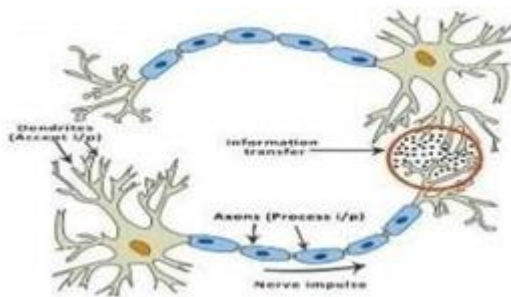


Fig: 4.1

The human brain is composed of 86 billion nerve cells called neurons. They are connected to other thousand cells by Axons. Stimuli from external environment or inputs from sensory organs are accepted by dendrites. These inputs create electric impulses, which quickly travel through the neural network. A neuron can then send

the message to other neuron to handle the issue or does not send it forward.

ANNs are composed of multiple nodes, which imitate biological neurons of human brain. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its activation or node value. Each link is associated with weight. ANNs are capable of learning, which takes place by altering weight values.

Neural network:

A neural network is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes. Thus a neural network is either a biological neural network, made up of real biological neurons, or an artificial neural network, for solving artificial intelligence (AI) problem. The connections of the biological neuron are modeled as weights. A positive weight reflects an excitatory connection, while negative values mean inhibitory connections. All inputs are modified by a weight and summed. This activity is referred as a linear combination. Finally, an activation function controls the amplitude of the output. For example, an acceptable range of output is usually between 0 and 1, or it could be - 1 and 1.

These artificial networks may be used for predictive modeling, adaptive control and applications where they can be trained via a dataset. Self-learning resulting from experience can occur within networks, which can derive conclusions from a complex and seemingly unrelated set of information.

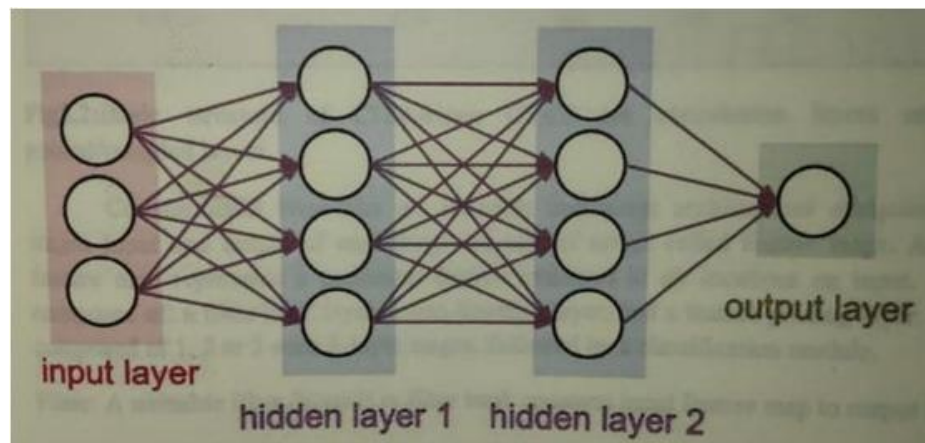


Fig. 4.2 A simple neural network

A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. The DNN finds the correct mathematical manipulation to turn the input into the output, whether it be a linear relationship or a non-linear relationship.

4.3.2 Convolutional Neural Networks

Convolutional Neural Networks are very similar to ordinary Neural Networks from the previous chapter: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity.

Convolutional Neural Networks (CNNs) are analogous to traditional ANNs in that they are comprised of neurons that self-optimize through learning. Each neuron will still receive an input and perform an operation (such as a scalar product followed by a non-linear function) - the basis of countless ANNs. From the input raw image vectors to the final output of the class score, the entire of the network will still be expressing a perceptual score function (the weight). The last layer will contain loss functions associated with the classes, and all of the regular tips and tricks developed for traditional ANNs still apply.

The only notable difference between CNNs and traditional ANNs is that CNNs are primarily used in the field of pattern recognition within images. This allows us to encode image-specific features into the architecture, making the network more suited for image-focused tasks-while further reducing the parameters required to setup the model. One of the largest limitations of traditional forms of ANN is that they tend to struggle with the computational complexity required to compute image data. Common machine learning benchmarking datasets such as the MNIST database of handwritten digits are suitable for most forms of ANN, due to its relatively small image dimensionality of just 28×28 . With this data set a single neuron in the first hidden layer will contain 784 weights ($28 \times 28 \times 1$ where 1 bare in mind that MNIST is normalised to just black and white values), which is manageable for most forms of ANN. If you consider a more substantial coloured image input of 64×64 , the number of weights on just a single neuron of the first layer increases substantially to 12,288. Also take into account that to deal with this scale of input, the network will also need to be a lot larger than one used to classify colour-normalised MNIST digits, then you will understand the drawbacks of using such models.

4.3.3 CNN Architecture

CNNs are feed forward networks in that information flow takes place in one direction only, from their inputs to their outputs. Just as artificial neural networks (ANN) are biologically inspired, so are CNNs. The visual cortex in the brain, which consists of alternating layers of simple and complex cells (Hubel

&Wiesel, 1959,1962), motivates their architecture.

CNN architectures come in several variations; however, in general, they consist of convolutional and pooling (or sub sampling) layers, which are grouped into modules. Either one or more fully connected layers, as in a standard feed forward neural network, follow these modules. Modules are often stacked on top of each other to form a deep model. It illustrates typical CNN architecture for a toy image classification task. An image is input directly to the network, and this is followed by several stages of convolution and pooling. Thereafter, representations from these operations feed one or more fully connected layers.

Finally, the last fully connected layer outputs the class label. Despite this being the most popular base architecture found in the literature, several architecture changes have been proposed in recent years with the objective of improving image classification accuracy or reducing computation costs. Although for the remainder of this section, we merely fleetingly introduce standard CNN architecture.

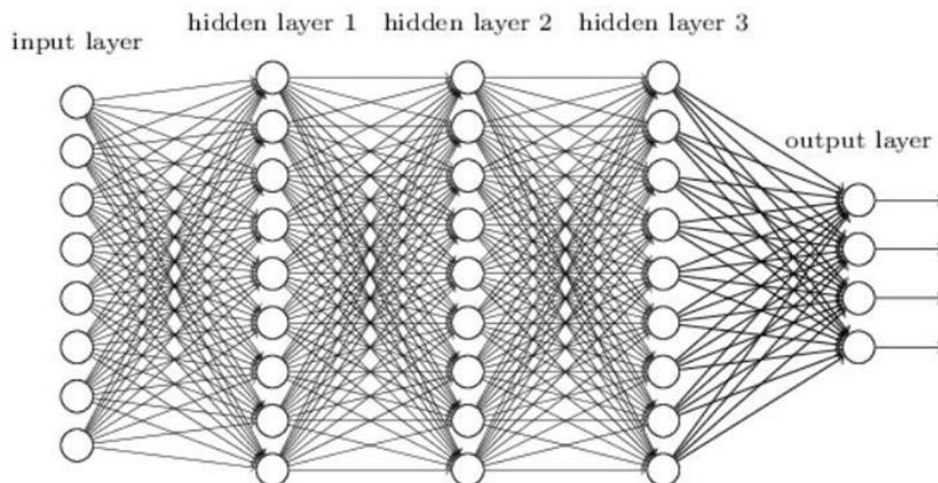


Fig: 4.3

4.3.4 Overall Architecture

CNNs are comprised of three types of layers. These are convolutional layers, pooling layers and fully- connected layers. When these layers are stacked, a CNN architecture has been formed. A simplified CNN architecture for MNIST classification is illustrated in Figure 2. input 0 9 convolution w/ReLU pooling output fully-connected w/ ReLu fully-connected ... Fig. 2: An simple CNN architecture, comprised of just five layers

The basic functionality of the example CNN above can be broken down into four key areas.

1. As found in other forms of ANN, the input layer will hold the pixel values of the image.
2. The convolutional layer will determine the output of neurons of which are connected to local regions of the input through the calculation of the scalar product between their weights and the region connected to the input volume. The rectified linear unit (commonly shortened to ReLu) aims to apply an 'elementwise' activation function such as sigmoid to the output of the activation produced by the previous layer.
3. The pooling layer will then simply perform downsampling along the spatial dimensionality of the given input, further reducing the number of parameters within that activation.
4. The fully-connected layers will then perform the same duties found in standard ANNs and attempt to produce class scores from the activations, to be used for classification.

It is also suggested that ReLu may be used between these layers, as to improve performance. Through this simple method of transformation, CNNs are able to transform the original input layer by layer using convolutional and downsampling techniques to produce class scores for classification and regression purposes. However, it is important to note that simply understanding the overall architecture of a CNN architecture will not suffice. The creation and optimisation of these models can take quite some time, and can be quite confusing. We will now explore in detail the individual layers, detailing their hyperparameters and connectivities.

4.3.5 Convolutional Layers

The convolutional layers serve as feature extractors, and thus they learn the feature representations of their input images. The neurons in the convolutional layers are arranged into feature maps. Each neuron in a feature map has a receptive field, which is connected to a neighborhood of neurons in the previous layer via a set of trainable weights, sometimes referred to as a filter bank. Inputs are convolved with the learned weights in order to compute a new feature map, and the convolved results are sent through a nonlinear activation function.

All neurons within a feature map have weights that are constrained to be equal; however, different feature maps within the same convolutional layer have different weights so that several features can be extracted at each location.

As the name implies, the convolutional layer plays a vital role in how CNNs operate. The layers parameters focus around the use of learnable kernels.

These kernels are usually small in spatial dimensionality, but spreads along the entirety of the depth of the input. When the data hits a convolutional layer, the layer convolves each filter across the spatial dimensionality of the input to produce a 2D activation map. These activation maps can be visualised.

As we glide through the input, the scalar product is calculated for each value in that kernel. From this the network will learn kernels that 'fire' when they see a specific feature at a given spatial position of the input. These are commonly known as activations.

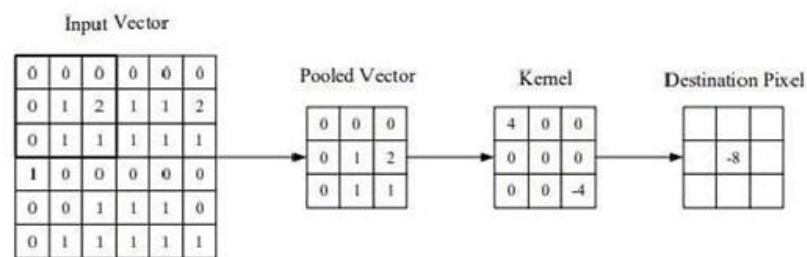


Fig: 4.4 Visual representation of a convolutional layers

The centre element of the kernel is placed over the input vector, of which is then calculated and replaced with a weighted sum of itself and any nearby pixels.

Every kernel will have a corresponding activation map, of which will be stacked along the depth dimension to form the full output volume from the convolutional layer.

As we alluded to earlier, training ANNs on inputs such as images results in models of which are too big to train effectively. This comes down to the fullyconnected manner of standard ANN neurons, so to mitigate against this every neuron in a convolutional layer is only connected to small region of the input volume. The dimensionality of this region is commonly referred to as the receptive field size of the neuron. The magnitude of the connectivity through the depth is nearly always equal to the depth of the input.

For example, if the input to the network is an image of size $64 \times 64 \times 3$

(aRGBcoloured image with a dimensionality of 64×64) and we set the receptive field size as 6×6 , we would have a total of 108 weights on each neuron within the convolutional layer. ($6 \times 6 \times 3$ where 3 is the magnitude of connectivity across the depth of the volume) To put this into perspective, a standard neuron seen in other forms of ANN would contain 12,288 weights each.

Convolutional layers are also able to significantly reduce the complexity of the model through the optimisation of its output. These are optimised through three hyperparameters, the depth, the stride and setting zero-padding.

The depth of the output volume produced by the convolutional layers can be manually set through the number of neurons within the layer to a the same region of the input.

This can be seen with other forms of ANNs, where the all of the neurons in the hidden layer are directly connected to every single neuron beforehand. Reducing this hyperparameter can significantly minimise the total number of neurons of the network, but it can also significantly reduce the pattern recognition capabilities of the model.

We are also able to define the stride in which we set the depth around the spatial dimensionality of the input in order to place the receptive field. For example if we were

to set a stride as 1, then we would have a heavily overlapped receptive field producing extremely large activations. Alternatively, setting the stride to a greater number will reduce the amount of overlapping and produce an output of lower spatial dimensions. Zero-padding is the simple process of padding the border of the input, and is an effective method to give further control as to the dimensionality of the output volumes. It is important to understand that through using these techniques, we will alter the spatial dimensionality of the convolutional layers output.

Despite our best efforts so far we will still find that our models are still enormous if we use an image input of any real dimensionality. However, methods have been developed as to greatly curtail the overall number of parameters within the convolutional layer.

Parameter sharing works on the assumption that if one region feature is useful to compute at a set spatial region, then it is likely to be useful in another region. If we constrain each individual activation map within the output volume to the same weights and bias, then we will see a massive reduction in the number of parameters being produced by the convolutional layer.

As a result of this as the backpropagation stage occurs, each neuron in the output will represent the overall gradient of which can be totalled across the depth - thus only updating a single set of weights, as opposed to every single one.

Pooling Layers

The purpose of the pooling layers is to reduce the spatial resolution of the feature maps and thus achieve spatial invariance to input distortions and translations. Initially, it was common practice to use average pooling aggregation layers to propagate the average of all the input values, of a small neighbourhood of an image to the next layer. However, in more recent models, max pooling aggregation layers propagate the maximum value within a receptive field to the next layer.

Pooling layers aim to gradually reduce the dimensionality of the representation, and thus further reduce the number of parameters and the computational complexity of the model.

The pooling layer operates over each activation map in the input, and scales its dimensionality using the “MAX” function. In most CNNs, these come in the form of max-pooling layers with kernels of a dimensionality of 2×2 applied with a stride of 2 along the spatial dimensions of the input. This scales the activation map down to 25% of the original size - whilst maintaining the depth volume to its standard size.

Due to the destructive nature of the pooling layer, there are only two generally observed methods of max-pooling. Usually, the stride and filters of the pooling layers are both set to 2×2 , which will allow the layer to extend through the entirety of the spatial dimensionality of the input. Furthermore overlapping pooling may be utilised, where the stride is set to 2 with a kernel size set to 3. Due to the destructive nature of pooling, having a kernel size above 3 will usually greatly decrease the performance of the model.

It is also important to understand that beyond max-pooling, CNN architectures may contain general- pooling. General pooling layers are comprised of pooling neurons that are able to perform a multitude of common operations including L1/L2-normalisation, and average pooling. However, this tutorial will primarily focus on the use of max-pooling.

Fully Connected Layers

Several convolutional and pooling layers are usually stacked on top of each other to extract more abstract feature representations in moving through the network. The fully connected layers that follow these layers interpret these feature representations and perform the function of high-level reasoning. . For classification problems, it is standard to use the softmax operator on top of a DCNN. While early success was enjoyed by using radial basis functions (RBFs), as the classifier on top

of the convolutional towers found that replacing the softmax operator with a support vector machine (SVM) leads to improved classification accuracy.

The fully-connected layer contains neurons of which are directly connected to the neurons in the two adjacent layers, without being connected to any layers within them. This is analogous to way that neurons are arranged in traditional forms of ANN.

Despite the relatively small number of layers required to form a CNN, there is no set way of formulating a CNN architecture. That being said, it would be idiotic to simply throw a few of layers together and expect it to work. Through reading of related literature it is obvious that much like other forms of ANNs, CNNs tend to follow a common architecture. This common architecture is illustrated in Figure 2, where convolutional layers are stacked, followed by pooling layers in a repeated manner before feeding forward to fully-connected layers.

Convolutional Neural Networks differ to other forms of Artificial Neural Network in that instead of focusing on the entirety of the problem domain, knowledge about the specific type of input is exploited. This in turn allows for a much simpler network architecture to be set up.

This paper has outlined the basic concepts of Convolutional Neural Networks, explaining the layers required to build one and detailing how best to structure the network in most image analysis tasks.

Research in the field of image analysis using neural networks has somewhat slowed in recent times. This is partly due to the incorrect belief surrounding the level of complexity and knowledge required to begin modelling these superbly powerful machine learning algorithms. The authors hope that this paper has in some way reduced this confusion, and made the field more accessible to beginners.

Training

CNNs and ANN in general use learning algorithms to adjust their free parameters in order to attain the desired network output. The most common algorithm used for this purpose is backpropagation. Backpropagation computes the gradient of an objective function to determine how to adjust a network's parameters in order to minimize errors that affect performance. A commonly experienced problem with training CNNs, and in particular DCNNs, is overfitting, which is poor performance on a held-out test set after the network is trained on a small or even large training set. This affects the model's ability to generalize on unseen data and is a major challenge for DCNNs that can be assuaged by regularization.

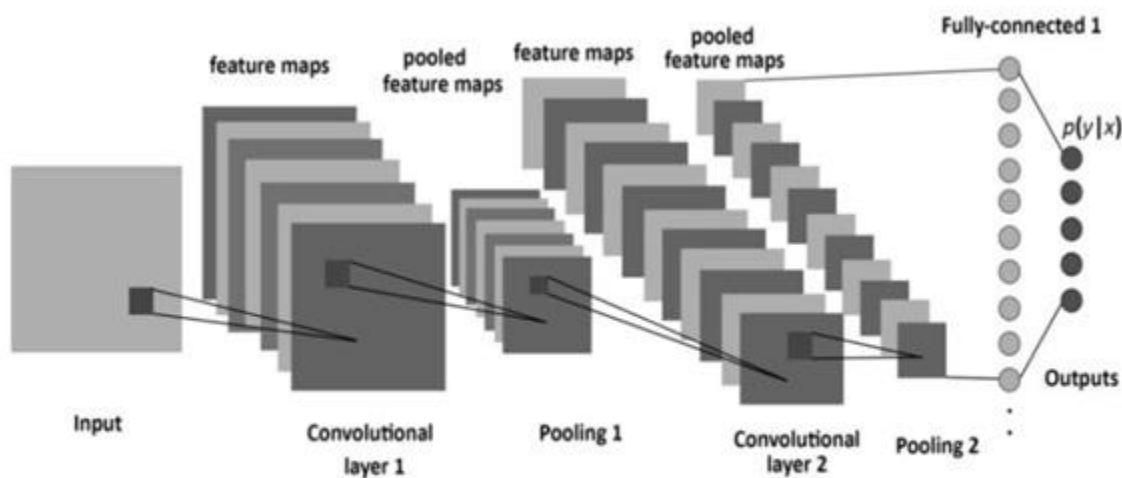


Fig: 4.5

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.

Types of Caffe Models

Open Pose

The first real-time multi-person system is portrayed by OpenPose which can collectively sight human body, hand, and facial keypoints (in total 130 keypoints) on single pictures.

Fully Convolutional Networks for Semantic Segmentation

In the absolutely convolutional networks (FCNs) Fully Convolutional Networks are the reference implementation of the models and code for the within the PAMI FCN and CVPR FCN papers.

Cnn-vis

Cnn-vis is an open-source tool that lets you use convolutional neural networks to generate images. It has taken inspiration from the Google's recent Inceptionism blog post.

Speech Recognition

Speech Recognition with the caffe deep learning framework.

DeconvNet

Learning Deconvolution Network for Semantic Segmentation.

Coupled Face Generation

This is the open source repository for the Coupled Generative Adversarial Network (CoupledGAN or CoGAN) work. These models are compatible with Caffe master, unlike earlier FCNs that required a pre-release branch (note: this reference edition of the models remains ongoing and not all of the models have yet been ported to master).

Codes for Fast Image Retrieval

To create the hash-like binary codes it provides effective framework for fast image retrieval.

SegNet and Bayesian SegNet

SegNet is real-time semantic segmentation architecture for scene understanding.

Deep Hand

It gives pre-trained CNN models.

DeepYeast

Deep Yeast may be an 11-layer convolutional neural network trained on biaural research pictures of yeast cells carrying fluorescent proteins with totally different subcellular localizations.

Python VS other languages for Object Detection: Object detection may be a domain-specific variation of the machine learning prediction drawback. Intel's OpenCV library that is implemented in C/C++ has its interfaces offered during a very vary of programming environments like C#, Matlab, Octave, R, Python and then on. Why Python codes are much better option than other language codes for object detection are more compact and readable code.

Python uses zero-based indexing. Dictionary (hashes) support provided.

Simple and elegant Object-oriented programming Free and open

Multiple functions can be package in one module

More choices in graphics packages and toolsets Supervised learning also plays an important role.

The utility of unsupervised pre-training is usually evaluated on the premise of what performance is achieved when supervised fine-tuning. This paper reviews and discusses the fundamentals of learning

as well as supervised learning for classification models, and also talks about the mini batch stochastic gradient descent algorithm that is used to fine-tune many of the models.

Object Classification in Moving Object Detection Object classification works on the shape, motion, color and texture. The classification can be done under various categories like plants, objects, animals, humans etc. The key concept of object classification is tracking objects and analysing their features.

Shape-Based

A mixture of image-based and scene based object parameters such as image blob (binary large object) area, the as pectration of blob bounding box and camera zoom is given as input to this detection system. Classification is performed on the basis of the blob at each and every frame. The results are kept in the histogram.

Motion-Based

When an easy image is given as an input with no objects in motion, this classification isn't required. In general, non-rigid articulated human motion shows a periodic property; therefore this has been used as a powerful clue for classification of moving objects. based on this useful clue, human motion is distinguished from different objects motion. ColorBased- though color isn't an applicable live alone for police investigation and following objects, but the low process value of the colour primarily based algorithms makes the coloura awfully smart feature to be exploited. As an example, the color-histogram based technique is employed for detection of vehicles in period. Color bar chart describes the colour distribution in a very given region that is powerful against partial occlusions.

Texture-Based

The texture-based approaches with the assistance of texture pattern recognition work just like motion-based approaches. It provides higher accuracy, by exploitation overlapping native distinction social control however might need longer, which may be improved exploitation some quick techniques. I. proposed WORK Authors have applied period object detection exploitation deep learning and OpenCV to figure to work with video streams and video files. This will be accomplished using the highly efficient open computer vision. Implementation of proposed strategy includes caffe-model based on Google Image Scenery; Caffe offers the model definitions, optimization settings, pre-trained weights[4].

Prerequisite includes Python 3.7, OpenCV 4 packages and numpy to complete this task of object detection. NumPy is the elementary package for scientific computing with Python. It contains among other things: a strong N-dimensional array object, subtle (broadcasting) functions tools for integrating C/C++ and fortran code, helpful linear algebra, Fourier transform, and random as well as supervised learning for classification models, and also talks about the mini batch stochastic gradient descent algorithm that is used to fine-tune many of the models.

Object Classification in Moving Object Detection Object classification works on the shape, motion, color and texture. The classification can be done under various categories like plants, objects, animals, humans etc. The key concept of object classification is tracking objects and analysing their features.

Shape-Based

A mixture of image-based and scene based object parameters such as image blob (binary large object) area, the aspect ratio of blob bounding box and camera zoom is given as input to this detection system. Classification is performed on the basis of the blob at each and every frame. The results are kept in the histogram.

Motion-Based

When an easy image is given as an input with no objects in motion, this classification isn't required. In general, non-rigid articulated human motion shows a periodic property; therefore this has been used as a powerful clue for classification of moving objects. based on this useful clue, human motion is distinguished from different objects motion. ColorBased- though color isn't an applicable live alone

for police investigation and following objects, but the low process value of the colour primarily based algorithms makes the colour a awfully smart feature to be exploited. As an example, the color- histogram based technique is employed for detection of vehicles in period. Color bar chart describes the colour distribution in a very given region that is powerful against partial occlusions.

Texture-Based

The texture-based approaches with the assistance of texture pattern recognition work just like motion- based approaches. It provides higher accuracy, by exploitation overlapping native distinction social control however might need longer, which may be improved exploitation some quick techniques. I. proposed WORK Authors have applied period object detection exploitation deep learning and OpenCV to figure to work with video streams and video files. This will be accomplished using the highly efficient open computer vision. Implementation of proposed strategy includes caffe- model based on Google Image Scenery; Caffe offers the model definitions, optimization settings, pre- trained weights[4]. Prerequisite includes Python 3.7, OpenCV 4 packages and numpy to complete this task of object detection. NumPy is the elementary package for scientific computing with Python. It contains among other things: a strong N-dimensional array object, subtle (broadcasting) functions tools for integrating C/C++ and fortran code, helpful linear algebra, Fourier transform, and randomnumber capabilities. Numpy works in backend to provide statistical information of resemblance of object with the image scenery caffemodel database. Object clusters can be created according to fuzzy value provided by NumPy. This project can detect live objects from the videos and images.

LEARNING FEATURE HIERARCHY:

Learn hierarchy all the way from pixels classifier One layer extracts features from output of previous layer, train all layers jointly

Zero-One Loss

The models given in these deep learning tutorials are largely used for classification. The major aim of training a classifier is to reduce the amount of errors (zero-one loss) on unseen examples

Negative Log-Likelihood Loss

Optimizing it for large models (thousands or millions of parameters) is prohibitively expensive (computationally) because the zero-one loss isn't differentiable. In order to achieve this maximization of the log-likelihood is done on the classifier given all the labels in a training set [14]. The likelihood of the correct class and number of right predictions is not the equal, but they are pretty similar from the point of view of a randomly initialized classifier. As the likelihood and zero-one loss are different objectives but we should always see that they are correlated on the validation set but sometimes one will rise while the other falls, or vice-versa.

Stochastic Gradient Descent

Ordinary gradient descent is an easy rule within which we repeatedly create tiny steps downward on an error surface defined by a loss function of some parameters. For the aim of normal gradient descent we take into account that the training data is rolled into the loss function. Then the pseudo code of this algorithm can be represented as Stochastic gradient descent (SGD) works according to similar principles as random gradient descent (SGD) operates on the basis of similar

principles as normal gradient descent. It quickly proceeds by estimating the gradient from simply a few examples at a time instead of complete training set. In its purest kind, we use simply one example at a time to estimate the gradient.

Caffe is a deep learning framework or else we can say a library it's made with expression speed and modularity in mind they will put by Berkeley artificial intelligence research and created by young King Gia there are many deep learning or machine learning frameworks for computer vision like tensorflow ,Tiano, Charis and SVM[2]. But why exactly we implement edition cafe there as on is its expressive architecture we can easily switch between CPU and GPU while training on GPU machinemodules and optimization for Our problem is defined by configuration without hard coding. It supports extensible code since cafes are open source library. It is four foot by over twenty thous and developers and github since its birth it offers coding platform in extensible languages like Python and C++. The next reason is speed for training the neural networks speed is the primary constraint. Caffe can process over million images in a single day with the standard media GPU that is milliseconds per image. Whereas the same dataset of million images can take weeks for Tiana and Kara's Caffe is the fastest convolution neural network present community as mentioned earlier since its open source library huge number of research arepowered by cafe and every single day something new is coming out of it.

4.4:OPENCPUTERVISION

4.4.1 Introduction

OpenCV stands for Open supply pc Vision Library is associate open supply pc 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 becomes very easy for businesses to utilize and modify the code with OpenCV as it is a BSD-licensed product. It is a rich wholesome libraby 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 is used for various functions such as discover and acknowledging faces. Identify objects classify human actions. In videos, track camera movements, track moving objects. Extract 3D models of objects, manufacture 3D purpose clouds from stereo cameras, sew pictures along to provide a high-resolution image of a complete scene, find similar pictures from a picture information, remove red eyes from images that are clicked with the flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality.

Officially launched in 1999 the OpenCV project was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects including real-time ray tracing and 3D display walls The main contributors to the project included a number of optimization experts in Intel Russia, as well as Intel's Performance Library Team. In the early days of OpenCV, the goals of the project were describedas:

- Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel.

- Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.
- Advance vision-based commercial applications by making portable, performance-optimized code available for free – with a license that did not require code to be open or free itself.

The first alpha version of OpenCV was released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and five betas were released between 2001 and 2005. The first

1.0 version was released in 2006. A version 1.1 "pre-release" was released in October 2008.

The second major release of the OpenCV was in October 2009. OpenCV 2 includes major changes to the C++ interface, aiming at easier, more type-safe patterns, new functions, and better implementations for existing ones in terms of performance (especially on multi-core systems). Official releases now occur every six months and development is now done by an independent Russian team supported by commercial corporations.

In August 2012, support for OpenCV was taken over by a non-profit foundation OpenCV.org, which maintains a developer and user site.

On May 2016, Intel signed an agreement to acquire ITSEEZ, a leading developer of OpenCV.

OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real- time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source BSD license.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now.

There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

4.4.2 Applications

- ☐ 2D and 3D feature toolkits
- ☐ Egomotion estimation
- ☐ Facial recognition system
- ☐ Gesture recognition
- ☐ Human–computer interaction (HCI)
- ☐ Mobile robotics
- ☐ Motion understanding
- ☐ Object identification
- ☐ Segmentation and recognition
- ☐ Stereopsis stereo vision: depth perception from 2 cameras
- ☐ Structure from motion (SFM)
- ☐ Motion tracking
- ☐ Augmented reality

To support some of the above areas, OpenCV includes a statistical machine learning library that contains :

- ☐ Boosting Decision tree learning
- ☐ Gradient boosting trees
- ☐ Expectation-maximization algorithm
- ☐ k-nearestneighbor algorithm

- ☐ Naive Bayes classifier
- ☐ Artificial neural networks
- ☐ Random forest
- ☐ Random forest
- ☐ Support vector machine (SVM)
- ☐ Deep neural networks (DNN)

4.4.3 Libraries in OpenCV

NumPy is an acronym for "Numeric Python" or "Numerical Python". It is an open source extension module for Python, which provides fast precompiled functions for mathematical and numerical routines. Furthermore, NumPy enriches the programming language Python with powerful data structures for efficient computation of multi-dimensional arrays and matrices. The implementation is even aiming at huge matrices and arrays. Besides that the module supplies a large library of high-level mathematical functions to operate on these matrices and arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- ☐ A powerful N-dimensional array object
- ☐ Sophisticated (broadcasting) functions
- ☐ Tools for integrating C/C++ and Fortran code
- ☐ Useful linear algebra, Fourier Transform, and random number capabilities.

Numpy Array:

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

SciPy:

SciPy (Scientific Python) is often mentioned in the same breath with NumPy. SciPy extends the capabilities of NumPy with further useful functions for minimization, regression, Fourier- transformation and many others. NumPy is based on two earlier

Python modules dealing with arrays. One of these is Numeric. Numeric is like NumPy a Python module for high-performance, numeric computing, but it is obsolete nowadays. Another predecessor of NumPy is Numarray, which is a complete rewrite of Numeric but is deprecated as well. NumPy is a merger of those two, i.e. it is build on the code of Numeric and the features of Numarray.

The Python Alternative To Matlab :

Python in combination with Numpy, Scipy and Matplotlib can be used as a replacement for MATLAB. The combination of NumPy, SciPy and Matplotlib is a free (meaning both "free" as in "free beer" and "free" as in "freedom") alternative to MATLAB. Even though MATLAB has a huge number of additional toolboxes available, NumPy has the advantage that Python is a more modern and complete programming language and - as we have said already before - is open source. SciPy adds even more MATLAB-like functionalities to Python. Python is rounded out in the direction of MATLAB with the module Matplotlib, which provides MATLAB-like plotting functionality.

4.4.4 HaarCascadeClassifiers

The algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, haar features shown in below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle.

Now all possible sizes and locations of each kernel is used to calculate plenty of features. (Just imagine how much computation it needs? Even a 24x24 window results over 160000 features). For each feature calculation, we need to find sum of pixels under white and black rectangles. To solve this, they introduced the integral images. It

simplifies calculation of sum of pixels, how large may be the number of pixels, to an operation involving just four pixels. Nice, isn't it? It makes things super-fast.

But among all these features we calculated, most of them are irrelevant. For example, consider the image below. Top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applying on cheeks or any other place is irrelevant. So how do we select the best features out of 160000+ features? It is achieved by Adaboost.

For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. But obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that best classifies the face and non-face images. (The process is not as simple as this. Each image is given an equal weight in the beginning. After each classification, weights of misclassified images are increased. Then again same process is done. New error rates are calculated. Also new weights. The process is continued until required accuracy or error rate is achieved or required number of features are found).

Final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can't classify the image, but together with others forms a strong classifier. The paper says even 200 features provide detection with 95% accuracy. Their final setup had around 6000 features. (Imagine a reduction from 160000+ features to 6000 features. That is a big gain).

So now you take an image. Take each 24x24 window. Apply 6000 features to it. Check if it is face or not. Wow.. Wow..Isn't it a little inefficient and time consuming? Yes, it is. Authors have a good solution for that.

In an image, most of the image region is non-face region. So it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot. Don't process it again. Instead focus on region where there can be a face. This way, we can find more time to check a possible face region

For this they introduced the concept of Cascade of Classifiers. Instead of applying all the 6000 features on a window, group the features into different stages of classifiers and apply one-by-one. (Normally first few stages will contain very less number of features). If a window fails the first stage, discard it. We don't consider remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region. Haar-like features are digital image features used in object recognition. They owe their name to their intuitive similarity with Haar wavelets and were used in the first real-time face detector.

Historically, working with only image intensities (i.e., the RGB pixel values at each and every pixel of image) made the task of feature calculation computationally expensive. A publication by Papageorgiou et al. discussed working with an alternate feature set based on Haar wavelets instead of the usual image intensities. Paul Viola and Michael Jones adapted the idea of using Haar wavelets and developed the so-called Haar-like features. A Haar-like feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. This difference is then used to categorize subsections of an image. For example, with a human face, it is a common observation that among all faces the region of the eyes is darker than the region of the cheeks. Therefore, a common Haar feature for face detection is a set of two adjacent

rectangles that lie above the eye and the cheek region. The position of these rectangles is defined relative to a detection window that acts like a bounding box to the target object (the face in this case).

In the detection phase of the Viola–Jones object detection framework, a window of the target size is moved over the input image, and for each subsection of the image the Haar-like feature is calculated. This difference is then compared to a learned threshold that separates non-objects from objects. Because such a Haar-like feature is only a weak learner or classifier (its detection quality is slightly better than random guessing) a large number of Haar-like features are necessary to describe an object with sufficient accuracy. In the Viola–Jones object detection framework, the Haar-like features are therefore organized in something called a classifier cascade to form a strong learner or classifier.

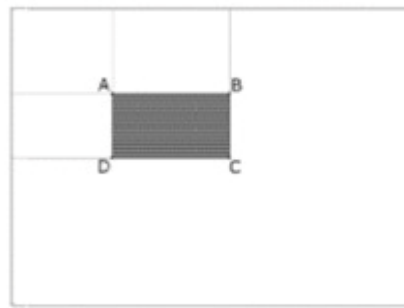
The key advantage of a Haar-like feature over most other features is its calculation speed. Due to the use of integral images, a Haar-like feature of any size can be calculated in constant time (approximately 60 microprocessor instructions for a 2-rectangle feature).

Rectangular Haar-like features

A simple rectangular Haar-like feature can be defined as the difference of the sum of pixels of areas inside the rectangle, which can be at any position and scale within the original image. This modified feature set is called 2-rectangle feature. Viola and Jones also defined 3-rectangle features and 4-rectangle features. The values indicate certain characteristics of a particular area of the image. Each feature type can indicate the existence (or absence) of certain characteristics in the image, such as edges or changes in texture. For example, a 2-rectangle feature can indicate where the border lies between a dark region and a light region.

Fast Computation of Haar-like features:

One of the contributions of Viola and Jones was to use summed-area tables, which they called integral images. Integral images can be defined as two-dimensional lookup tables in the form of a matrix with the same size of the original image. Each element of the integral image contains the sum of all pixels located on the up-left region of the original image (in relation to the element's position).



$$\text{Sum} = I(C) + I(A) - I(B) - I(D)$$

Fig: 5.1 2-Rectangle feature

OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:

- Core functionality (core) - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.
- Image Processing (imgproc) - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
- Video Analysis (video) - a video analysis module that includes motion estimation background subtraction, and object tracking algorithms.

- Camera Calibration and 3D Reconstruction (calib3d) - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
- 2D Features Framework (features2d) - salient feature detectors, descriptors, and descriptor matchers.
- Object Detection (objdetect) - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
- High-level GUI (highgui) - an easy-to-use interface to simple UI capabilities.
- Video I/O (videoio) - an easy-to-use interface to video capturing and video codecs.
- Some other helper modules, such as FLANN and Google test wrappers, Python bindings, and others.

CHAPTER 5

RESULTS AND DISCUSSION

INTRODUCTION TO IMPLEMENTATION OF PROBLEM

The Model

Deep learning is a popular technique used in computer vision. We chose Convolutional Neural Network (CNN) layers as building blocks to create our model architecture. CNNs are known to imitate how the human brain works when analyzing visuals.

A typical architecture of a convolutional neural network contain an input layer, some convolutional layers, some dense layers (aka. fully-connected layers), and an output layer . These are linearly stacked layers ordered in sequence.

Input Layer

The input layer has pre-determined, fixed dimensions, so the image must be pre-processed before it can be fed into the layer. We used OpenCV, a computer vision library, for object detection in the video.

The OpenCV contains pre-trained filters and uses Adaboost to quickly find and crop the object. The cropped object is then converted into gray scale using `cv2.cvtColor` and resized to 48-by-48 pixels with `cv2.resize`. This step greatly reduces the dimensions compared to the original RGB format with three colour dimensions (3, 48, 48). The pipeline ensures every image can be fed into the input layer as a (1, 48, 48) numpy array.

Convolutional Layers

The numpy array gets passed into the Convolution2D layer where we specify the

number of filters as one of the hyper parameters. The set of filters are unique with randomly generated weights. Each filter, (3, 3) receptive field, slides across the original image with shared weights to create a feature map.

Convolution generates feature maps that represent how pixel values are enhanced, for example, edge and pattern detection. A feature map is created by applying filter 1 across the entire image. Other filters are applied one after another creating a set of feature maps.

Pooling is a dimension reduction technique usually applied after one or several convolutional layers. It is an important step when building CNNs as adding more convolutional layers can greatly affect computational time. We used a popular pooling method called MaxPooling2D that uses (2, 2) windows across the feature map only keeping the maximum pixel value. The pooled pixels form an image with dimensions reduced by 4.

Dense Layers

The dense layer (aka fully connected layers), is inspired by the way neurons transmit signals through the brain. It takes a large number of input features and transform features through layers connected with trainable weights.

These weights are trained by forward propagation of training data then backward propagation of its errors. Back propagation starts from evaluating the difference between prediction and true value, and back calculates the weight adjustment needed to every layer before. We can control the training speed and the complexity of the architecture by tuning the hyper-parameters, such as learning rate and network density. As we feed in more data, the network is able to gradually make adjustments until errors are minimized. Essentially, the more layers/nodes we add to the network the better it can pick up signals.

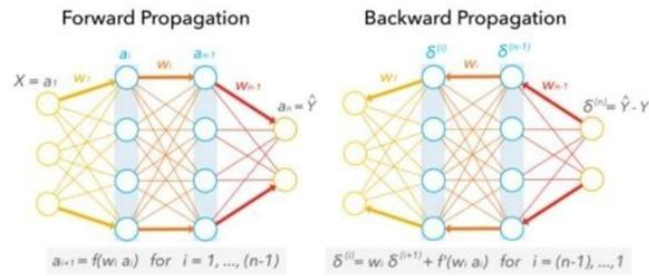


Fig: 6.1

As good as it may sound, the model also becomes increasingly prone to overfitting the training data. One method to prevent overfitting and generalize on unseen data is to apply dropout. Dropout randomly selects a portion (usually less than 50%) of nodes to set their weights to zero during training. This method can effectively control the model's sensitivity to noise during training while maintaining the necessary complexity of the architecture.

Output layer

The output layer in a CNN as mentioned previously is a fully connected layer, where the input from the other layers is flattened and sent so as to transform the output into the number of classes as desired by the network.

6. IMPLEMENTATION SNAPSHOTS OF SOURCE CODE

```
In [5]: import tkinter as tk
from PIL import Image, ImageTk
import subprocess

def Live():
    subprocess.call(['python', r'Livecam_Detect.py'])

def Video():
    subprocess.call(['python', r'Video_Detect.py'])

def Images():
    subprocess.call(['python', r'Image_Detect.py'])

Main_frame = tk.Tk()
Main_frame.geometry("1235x725")
Main_frame.resizable(False, False)
Main_frame.title("OBJECT DETECTION")
Main_frame.configure(background = 'black')

Main_frame.grid_rowconfigure(0, weight = 1)
Main_frame.grid_columnconfigure(0, weight = 1)

Bg_image = Image.open(r"C:\Users\praka\Desktop\Minor_project\MainBG2.png")
Bg_image_p = ImageTk.PhotoImage(Bg_image)
Bg_label = tk.Label(Main_frame, image = Bg_image_p)
Bg_label.place(x=0, y=0)

sub1img = Image.open(r"C:\Users\praka\Desktop\Minor_project\wedcam.png")
sub1img_p = ImageTk.PhotoImage(sub1img)

sub1button = tk.Button(Main_frame, command = Live, image = sub1img_p, bd=0, bg='#F6F4F4', activebackground='white')
sub1button.place(x=700, y=150)

Main_frame.grid_columnconfigure(0, weight = 1)

Bg_image = Image.open(r"C:\Users\praka\Desktop\Minor_project\MainBG2.png")
Bg_image_p = ImageTk.PhotoImage(Bg_image)
Bg_label = tk.Label(Main_frame, image = Bg_image_p)
Bg_label.place(x=0, y=0)

sub1img = Image.open(r"C:\Users\praka\Desktop\Minor_project\wedcam.png")
sub1img_p = ImageTk.PhotoImage(sub1img)

sub1button = tk.Button(Main_frame, command = Live, image = sub1img_p, bd=0, bg='#F6F4F4', activebackground='white')
sub1button.place(x=700, y=150)
sub1button_label = tk.Label(Main_frame, text = "Web Cam", fg="black", bg="#0d488f", font=('Gabriola', 20))
sub1button_label.place(x=709, y=252)

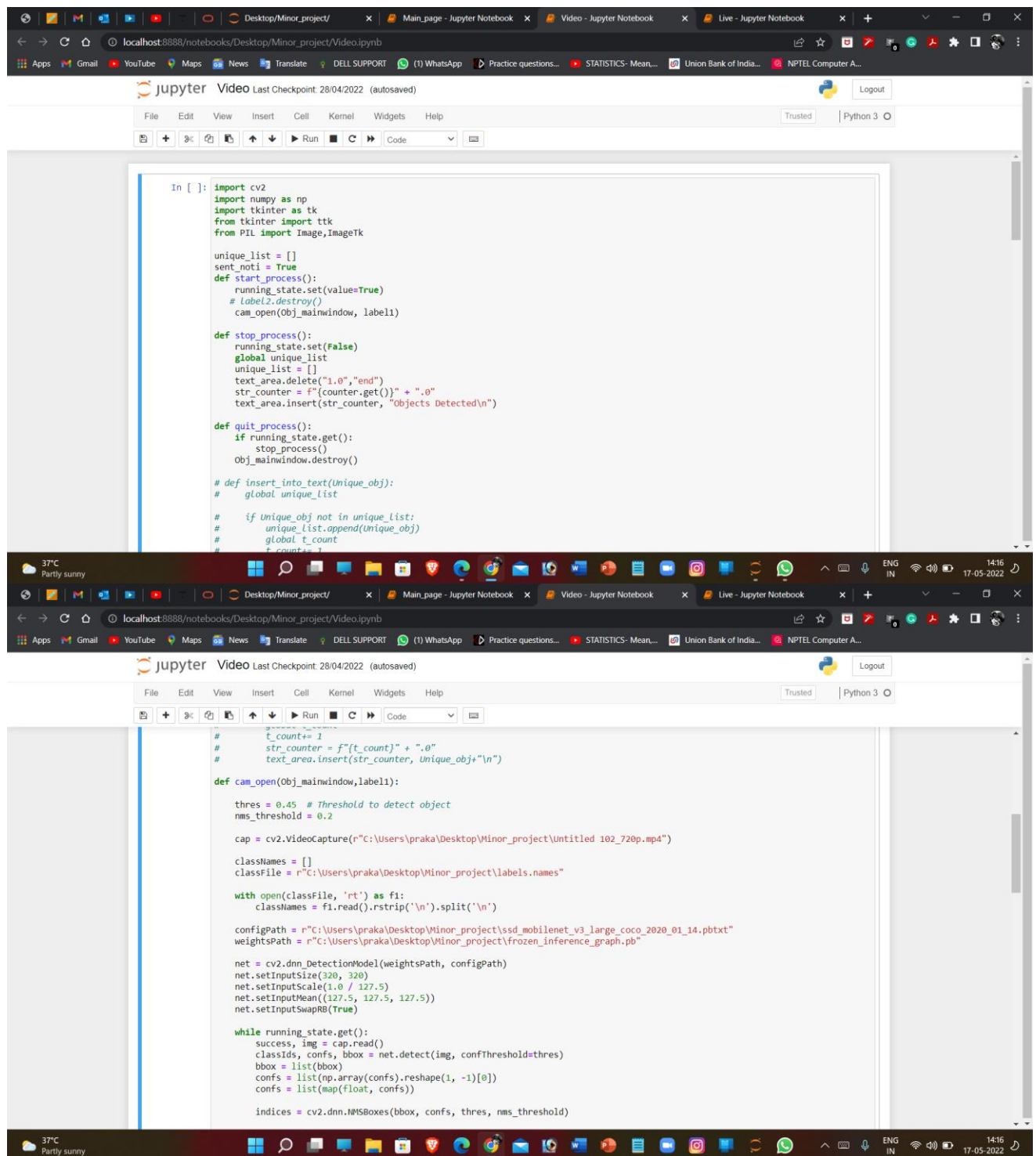
sub2img = Image.open(r"C:\Users\praka\Desktop\Minor_project\icon2 resised.png")
sub2img_p = ImageTk.PhotoImage(sub2img)

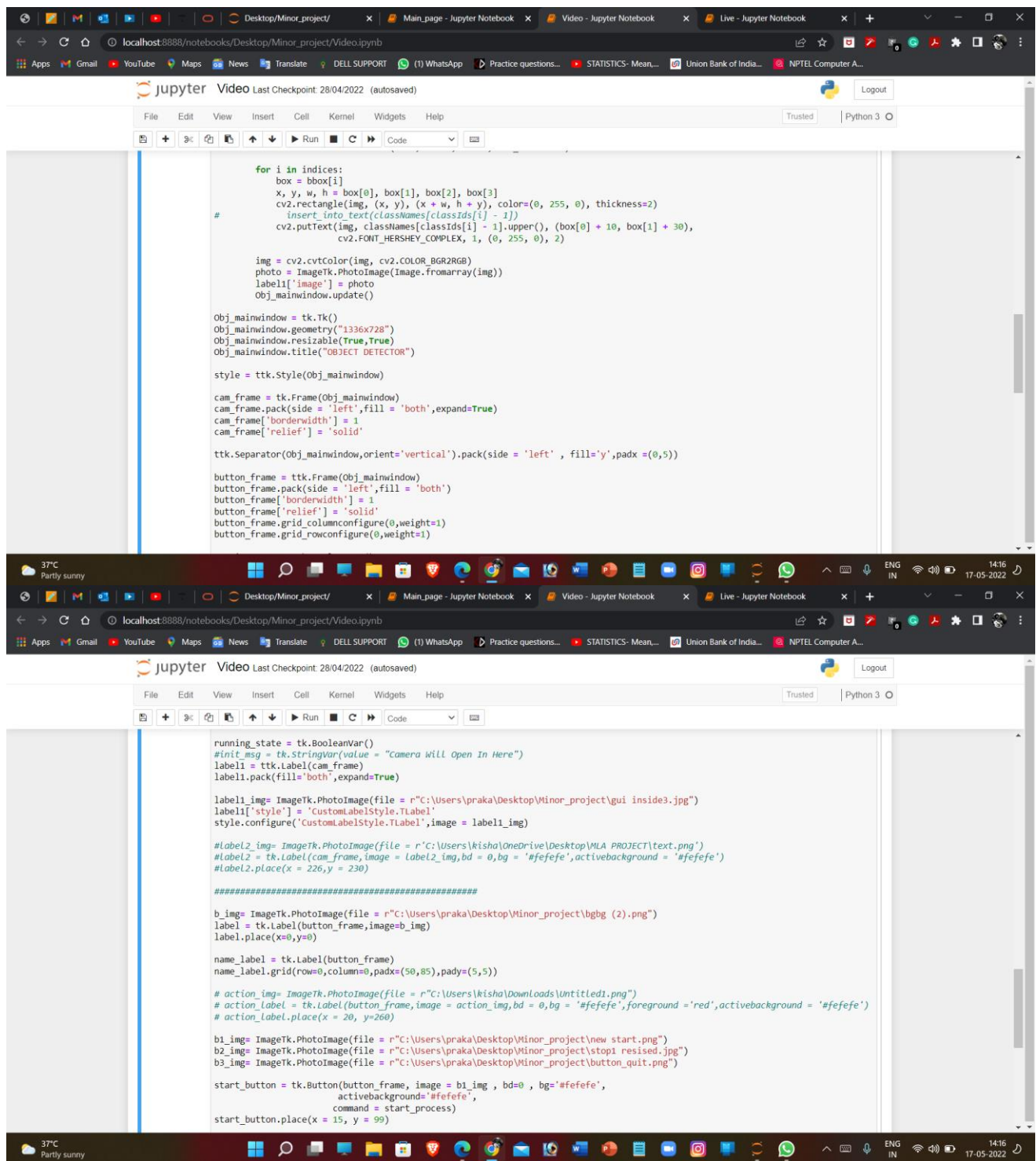
sub2button = tk.Button(Main_frame, command = Video, image = sub2img_p, bd=0, bg='#F6F4F4', activebackground='white')
sub2button.place(x=700, y=350)
sub2button_label = tk.Label(Main_frame, text = "Video", fg="black", bg="#0d488f", font=('Gabriola', 20))
sub2button_label.place(x=720, y=432)

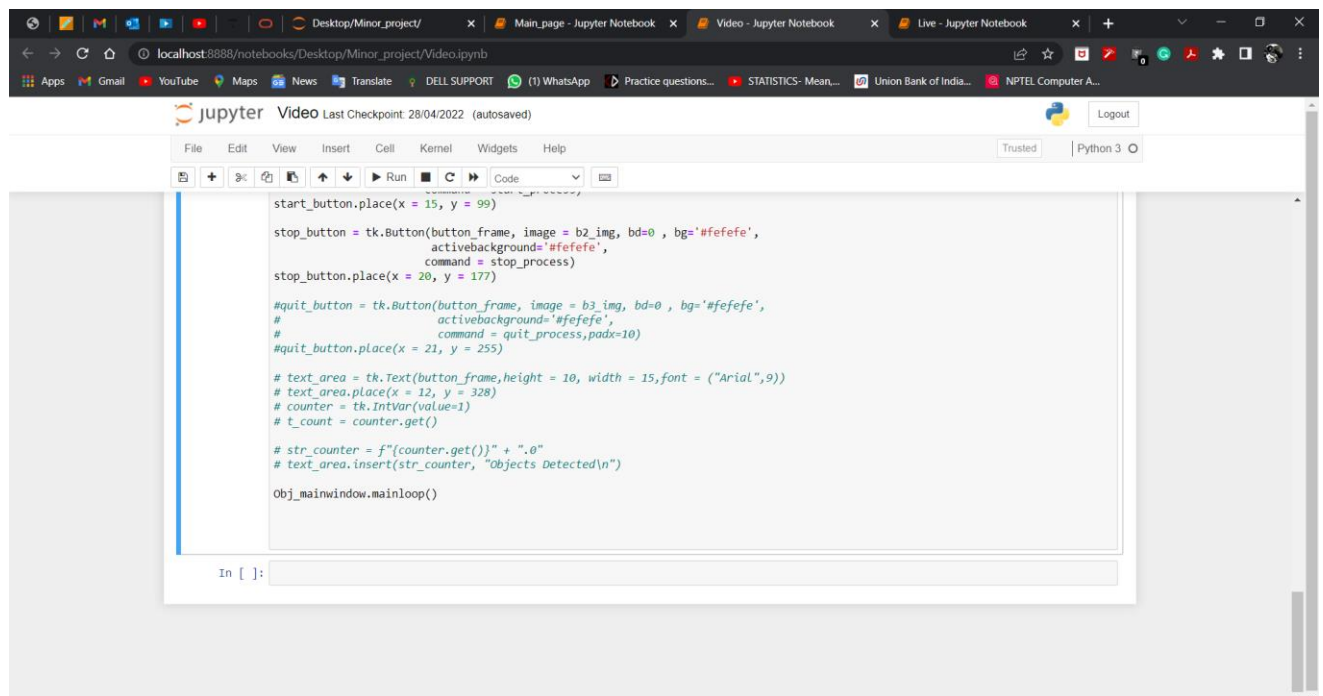
# sub3img = Image.open(r"C:\Users\kisha\Downloads\IMAGE 6.1 (1).jpg")
# sub3img_p = ImageTk.PhotoImage(sub3img)

# sub3button = tk.Button(Main_frame, command = Images, image = sub3img_p, bd=0, bg='#F6F4F4', activebackground='white')
# sub3button.place(x=700, y=550)
# sub3button_label = tk.Label(Main_frame, text = "Image", fg="black", bg="#0d488f", font=('Gabriola', 20))
# sub3button_label.place(x=720, y=632)

Main_frame.mainloop()
```







```
start_button.place(x = 15, y = 99)

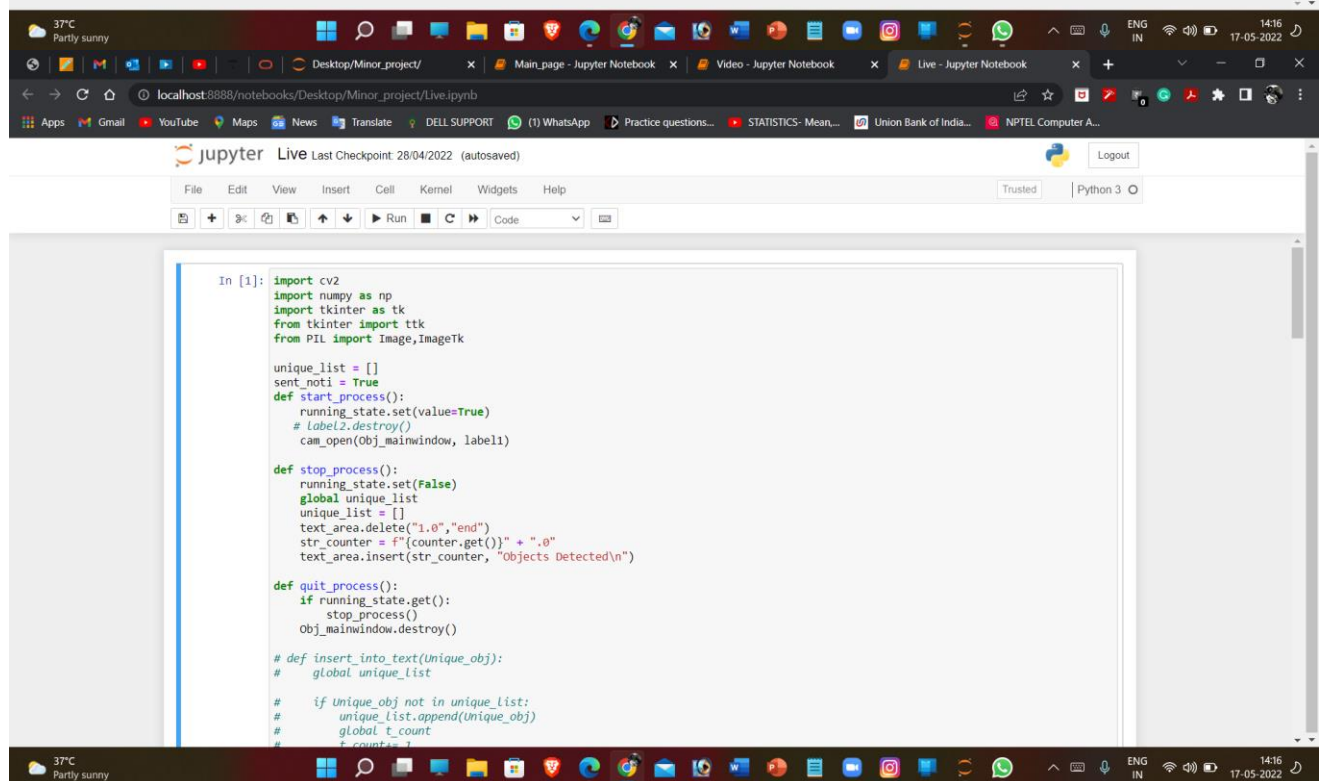
stop_button = tk.Button(button_frame, image = b2_img, bd=0 , bg='#fefe',
                        activebackground='#fefe',
                        command = stop_process)
stop_button.place(x = 20, y = 177)

#quit_button = tk.Button(button_frame, image = b3_img, bd=0 , bg='#fefe',
#                        activebackground='#fefe',
#                        command = quit_process,padx=10)
#quit_button.place(x = 21, y = 255)

# text_area = tk.Text(button_frame,height = 10, width = 15,font = ("Arial",9))
# text_area.place(x = 12, y = 328)
# counter = tk.IntVar(value=1)
# t_count = counter.get()

# str_counter = f"{counter.get()}" + ".0"
# text_area.insert(str_counter, "Objects Detected\n")

Obj_mainwindow.mainloop()
```



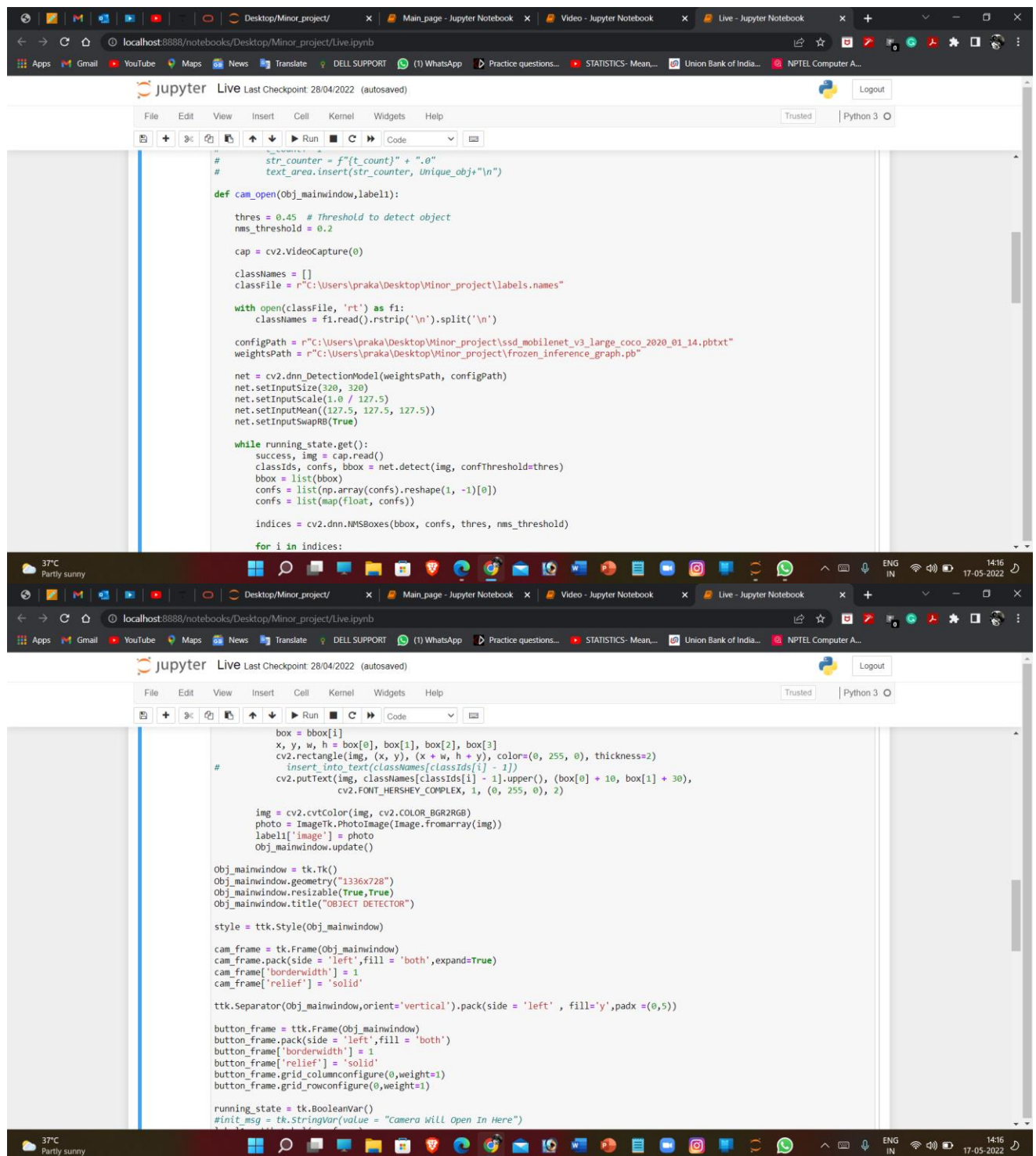
```
In [1]: import cv2
import numpy as np
import tkinter as tk
from tkinter import ttk
from PIL import Image,ImageTk

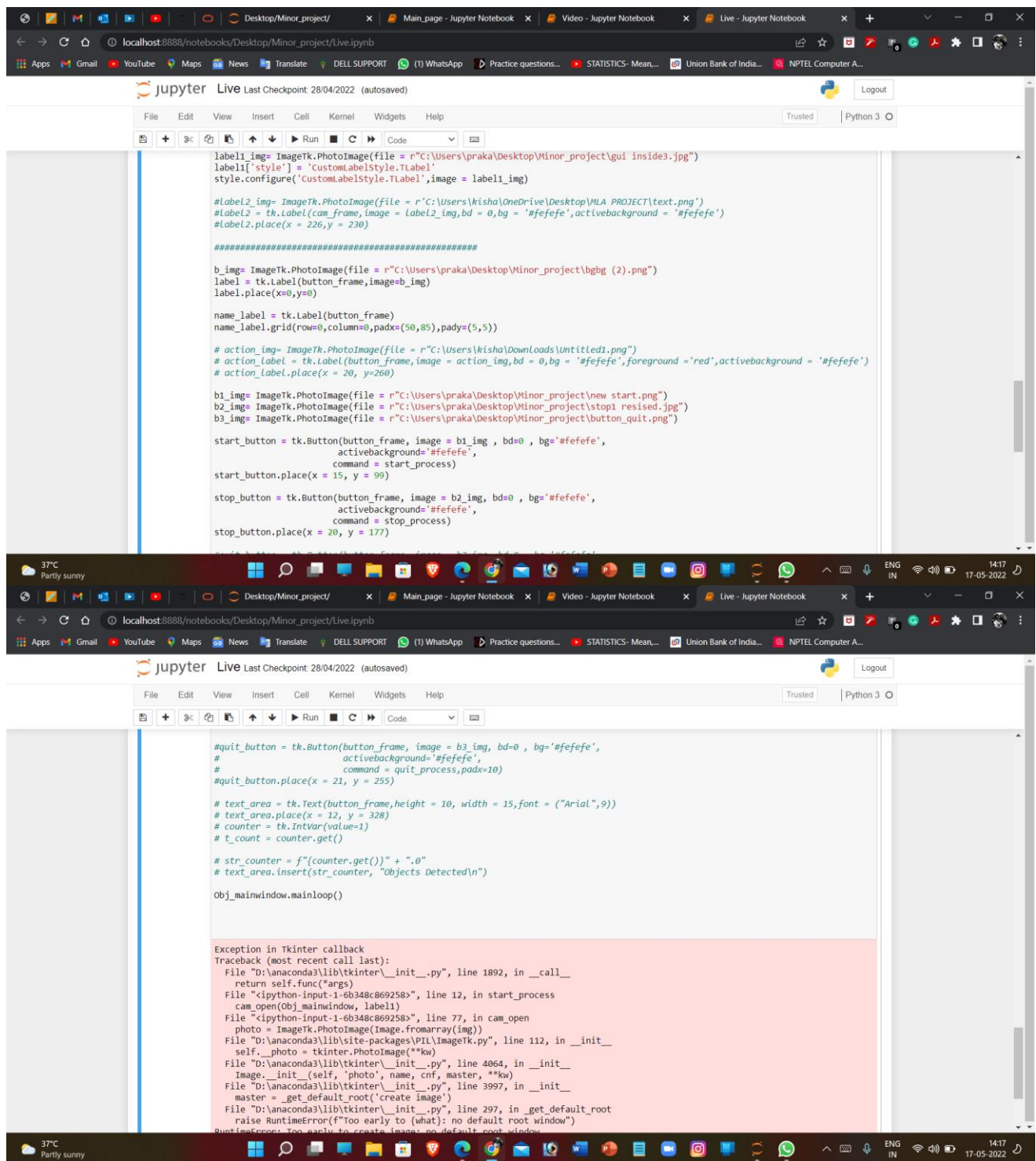
unique_list = []
sent_noti = True
def start_process():
    running_state.set(value=True)
    # Label2.destroy()
    cam_open(Obj_mainwindow, label1)

def stop_process():
    running_state.set(False)
    global unique_list
    unique_list = []
    text_area.delete("1.0","end")
    str_counter = f"{counter.get()}" + ".0"
    text_area.insert(str_counter, "Objects Detected\n")

def quit_process():
    if running_state.get():
        stop_process()
        Obj_mainwindow.destroy()

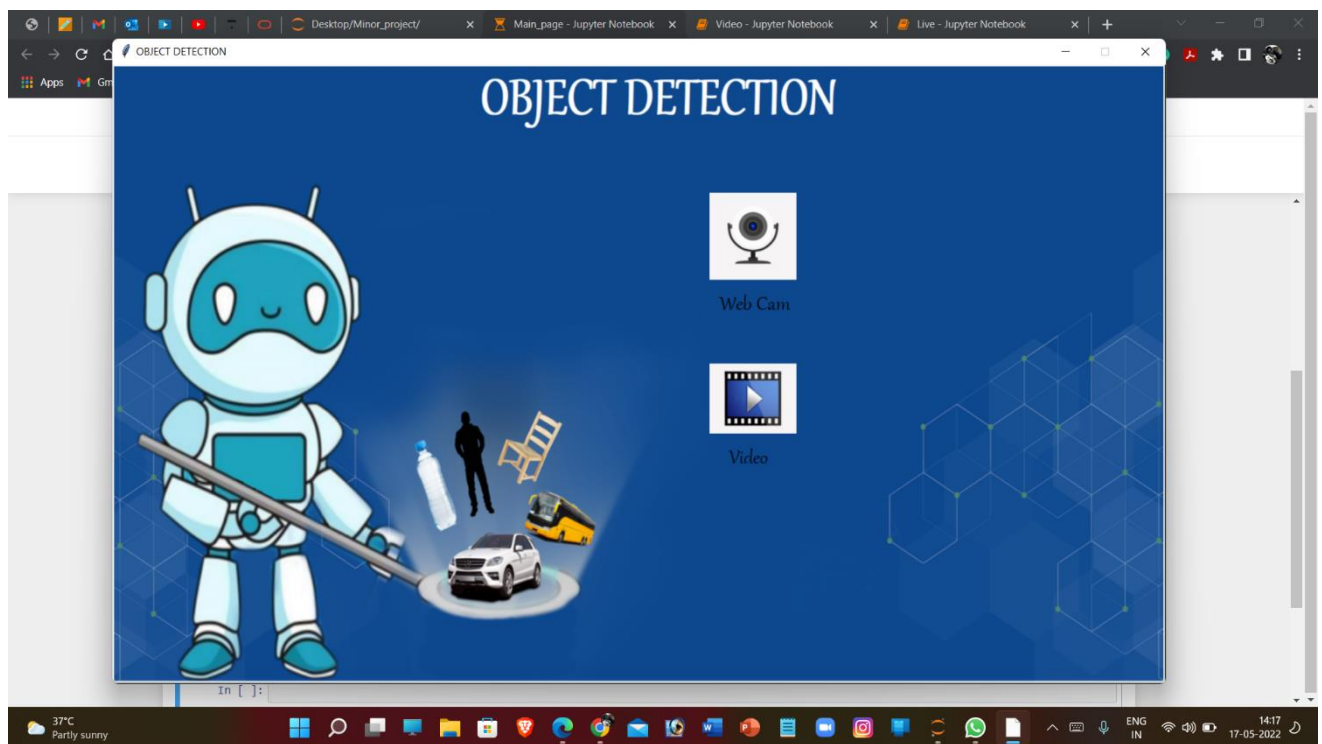
# def insert_into_text(unique_obj):
#     global unique_list
#     if unique_obj not in unique_list:
#         unique_list.append(unique_obj)
#         global t_count
#         t_count = t_count + 1
```

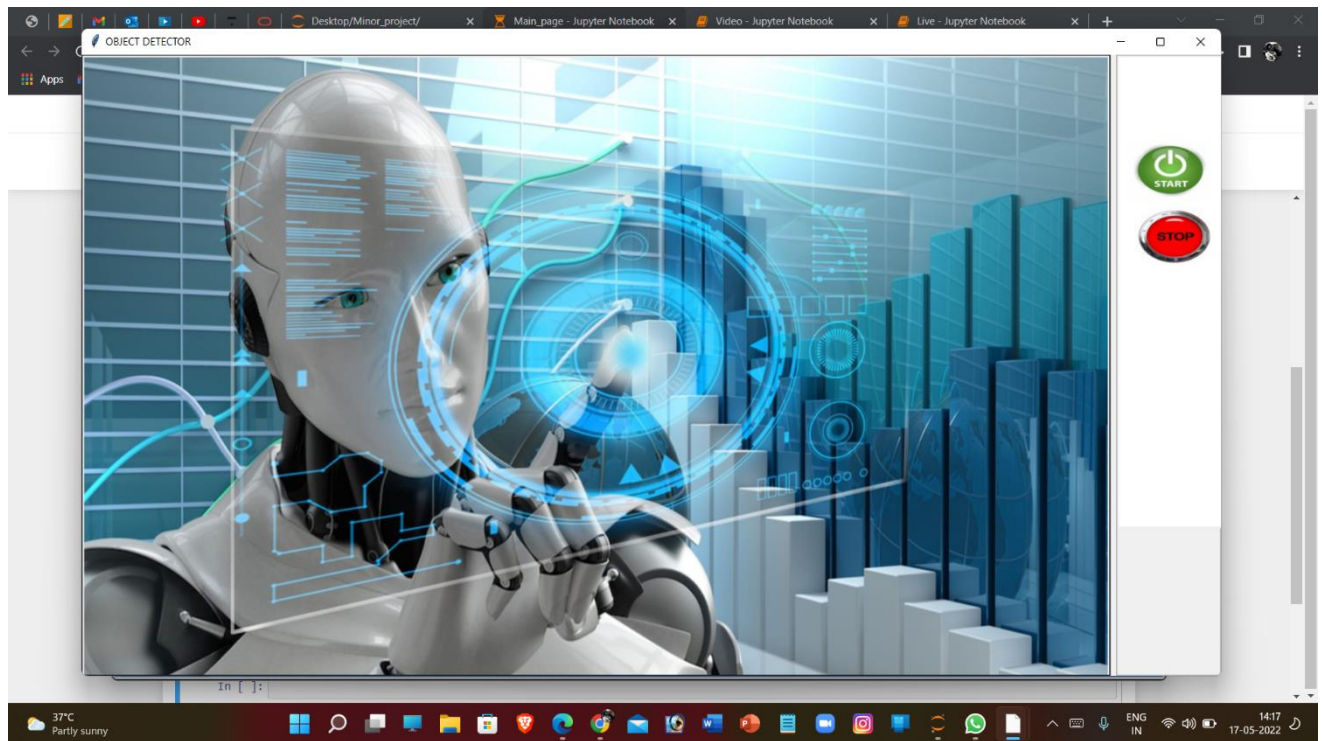


RESULTS:

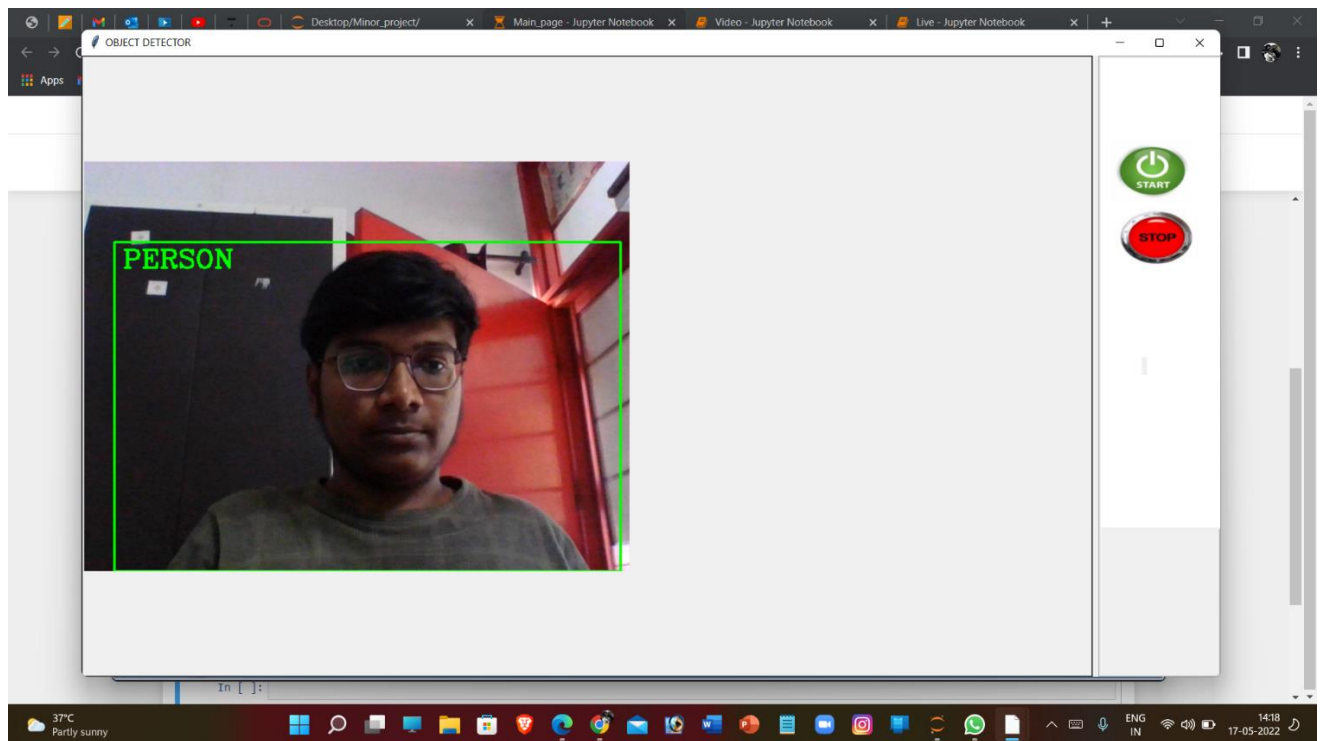
Our Project mainly deals with the detection aspect of the given moving inputs either in the form of videos from existing sources otherwise in the form of live input from closed circuit television present in almost all the places with basic security requirements or web cameras from computer users. The aim of the paper includes to segment the detection in different types and study on their characteristics like the required computational power etc.



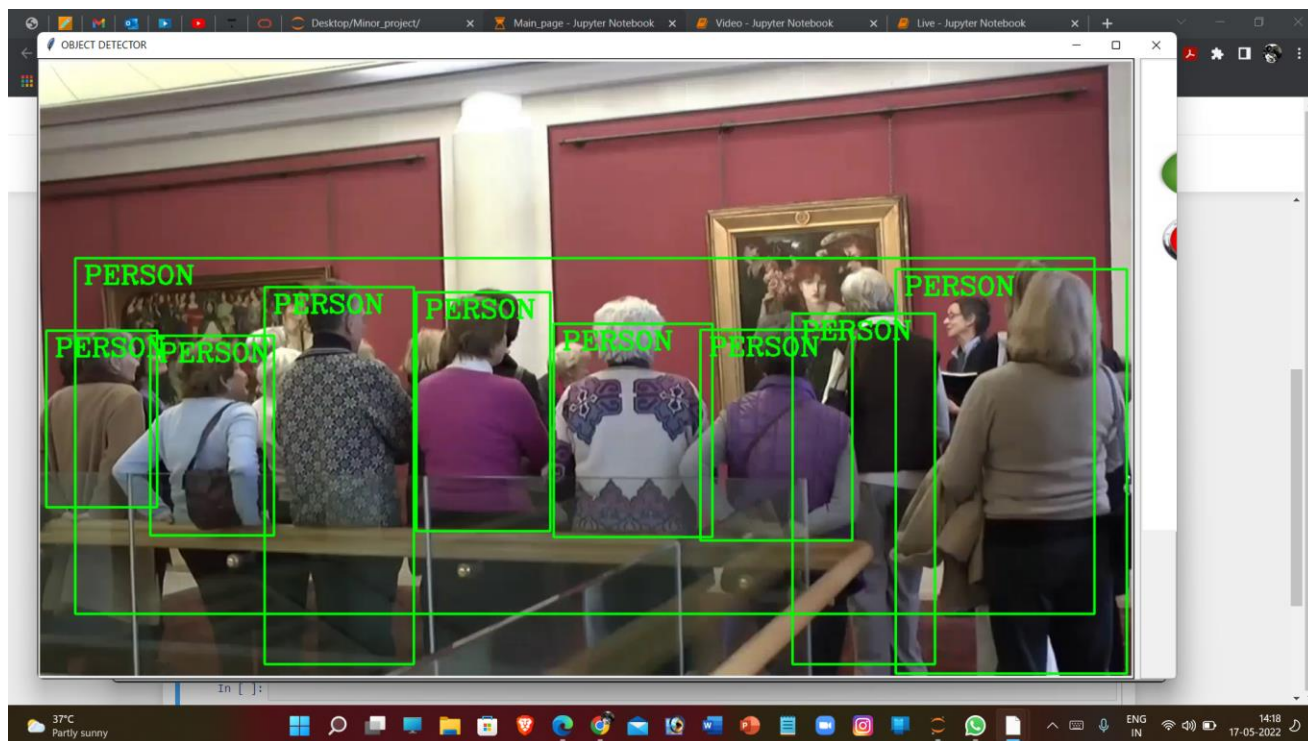
Main Menu consisting of graphical user interface



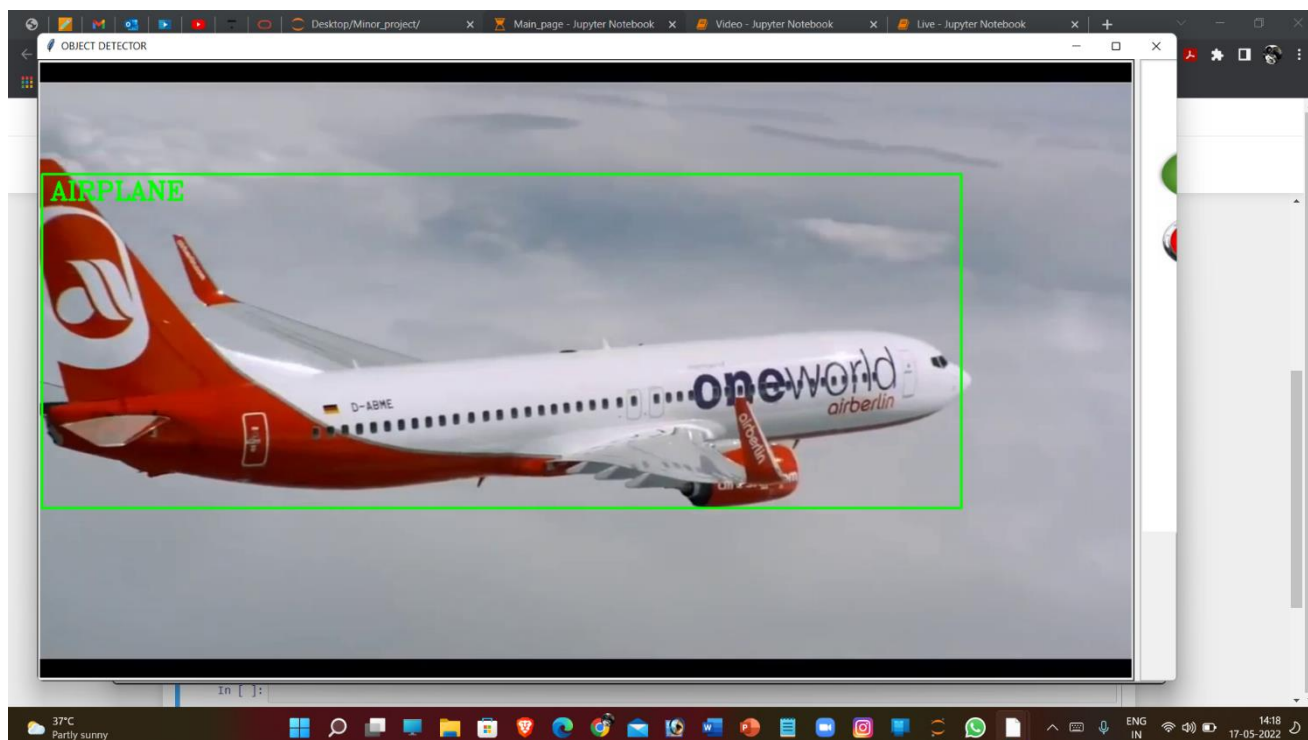
Start Stop screen for better Interaction with application.

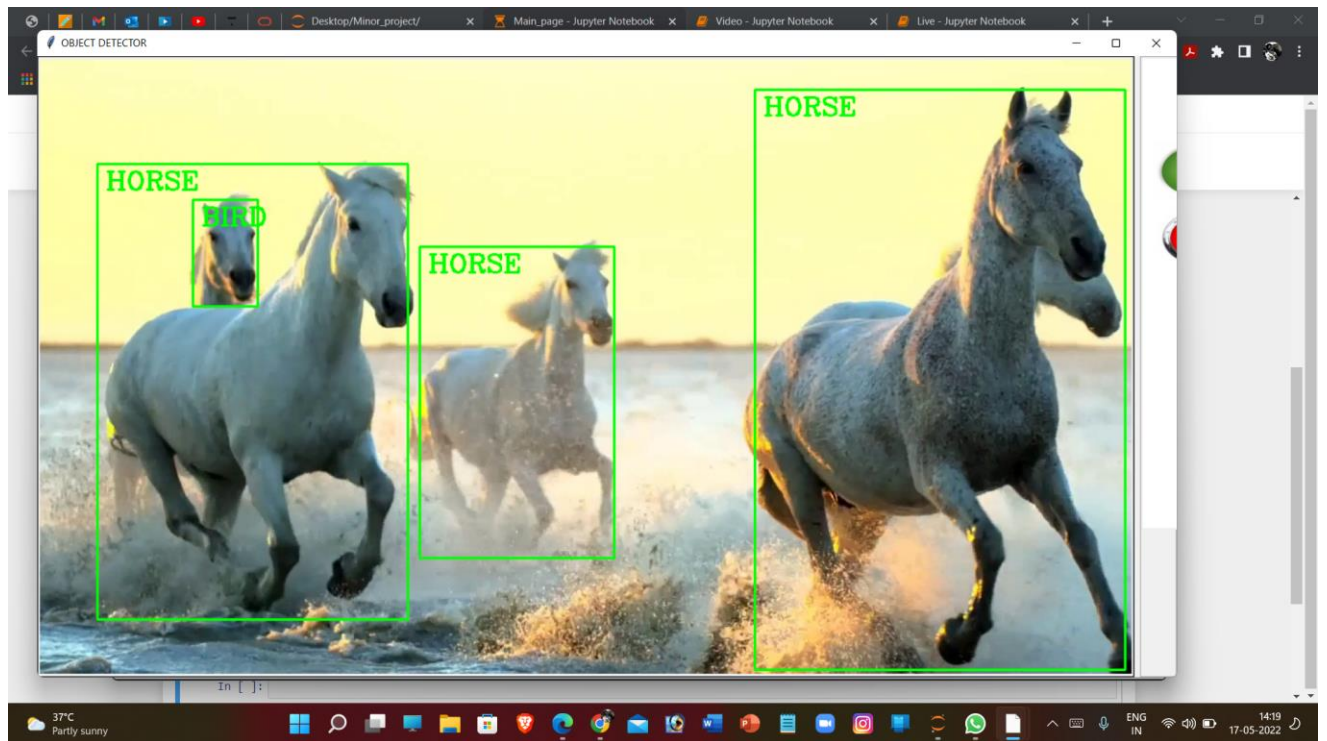


Object Detection with Live video from webcam as an input



Object detection taking place with video as input





7. CONCLUSIONS

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. 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 6-8 FPS.

By using this thesis and based on experimental results we are able to detect object more precisely and identify the objects individually with exact location of an object in the picture in x,y axis. This paper also provides experimental results on different methods for object detection and identification and compares each method for their efficiencies.

8. FUTURE ENHANCEMENTS

The object recognition system can be applied in the area of surveillance system, face recognition, fault detection, character recognition etc. The objective of this thesis is to develop an object recognition system to recognize the 2D and 3D objects in the image. The performance of the object recognition system depends on the features used and the classifier employed for recognition. This research work attempts to propose a novel feature extraction method for extracting global features and obtaining local features from the region of interest. Also the research work attempts to hybrid the traditional classifiers to recognize the object. The object recognition system developed in this research was tested with the benchmark datasets like COIL100, Caltech 101, ETH80 and MNIST. The object recognition system is implemented in MATLAB 7.5

It is important to mention the difficulties observed during the experimentation of the object recognition system due to several features present in the image. The research work suggests that the image is to be preprocessed and reduced to a size of 128 x 128. The proposed feature extraction method helps to select the important feature. To improve the efficiency of the classifier, the number of features should be less in number. Specifically, the contributions towards this research work are as follows,

An object recognition system is developed, that recognizes the two-dimensional and three-dimensional objects.

- The feature extracted is sufficient for recognizing the object and marking the location of the object. x The proposed classifier is able to recognize the object in less computational cost.
- The proposed global feature extraction requires less time, compared to the traditional feature extraction method.

- The performance of the SVM-kNN is greater and promising when compared with the BPN and SVM.
- The performance of the One-against-One classifier is efficient.
- Global feature extracted from the local parts of the image.
- Local feature PCA-SIFT is computed from the blobs detected by the Hessian-Laplace detector.
- Along with the local features, the width and height of the object computed through projection method is used.

The methods presented for feature extraction and recognition are common and can be applied to any application that is relevant to object recognition.

The proposed object recognition method combines the state-of-art classifier SVM and k-NN to recognize the objects in the image. The multiclass SVM is used to hybridize with the k-NN for the recognition. The feature extraction method proposed in this research work is efficient and provides unique information for the classifier.

The image is segmented into 16 parts, from each part the Hu's Moment invariant is computed and it is converted into Eigen component. The local feature of the image is obtained by using the Hessian-Laplace detector. This helps to obtain the objects feature easily and mark the object location without much difficulty.

As a scope for future enhancement,

- Features either the local or global used for recognition can be increased, to increase the efficiency of the object recognition system.
- Geometric properties of the image can be included in the feature vector for recognition.

- Using unsupervised classifier instead of a supervised classifier for recognition of the object.
- The proposed object recognition system uses grey-scale image and discards the color information. The color information in the image can be used for recognition of the object. Color based object recognition plays vital role in Robotics. Although the visual tracking algorithm proposed here is robust in many of the conditions, it can be made more robust by eliminating some of the limitations as listed below:
- In the Single Visual tracking, the size of the template remains fixed for tracking. If the size of the object reduces with the time, the background becomes more dominant than the object being tracked. In this case the object may not be tracked.
- Fully occluded object cannot be tracked and considered as a new object in the next frame.
- Foreground object extraction depends on the binary segmentation which is carried out by applying threshold techniques. So blob extraction and tracking depends on the threshold value.
- Splitting and merging cannot be handled very well in all conditions using the single camera due to the loss of information of a 3D object projection in 2D images.
- For Night time visual tracking, night vision mode should be available as an inbuilt feature in the CCTV camera.

To make the system fully automatic and also to overcome the above limitations, in future, multi-view tracking can be implemented using multiple cameras. Multi view tracking has the obvious advantage over single view tracking because of wide coverage

range with different viewing angles for the objects to be tracked.

In this thesis, an effort has been made to develop an algorithm to provide the base for future applications such as listed below.

- In this research work, the object Identification and Visual Tracking has been done through the use of ordinary camera. The concept is well extendable in applications like Intelligent Robots, Automatic Guided Vehicles, Enhancement of Security Systems to detect the suspicious behavior along with detection of weapons, identify the suspicious movements of enemies on boarders with the help of night vision cameras and many such applications.
- In the proposed method, background subtraction technique has been used that is simple and fast. This technique is applicable where there is no movement of camera. For robotic application or automated vehicle assistance system, due to the movement of camera, backgrounds are continuously changing leading to implementation of some different segmentation techniques like single Gaussian mixture or multiple Gaussian mixture models.
- Object identification task with motion estimation needs to be fast enough to be implemented for the real time system. Still there is a scope for developing faster algorithms for object identification. Such algorithms can be implemented using FPGA or CPLD for fast execution.

9. REFERENCES

- [1] Manjula S, Lakshmi Tamilselvan (2016) A study on object detection.
- [2] Jun Deng, Xiaojing Xuan, Weifeng Wang, Zhao Li, Hanwen Yao, Zhiqiang Wang (2020) A review of research on object detection based on deep learning.
- [3] Ajeet Ram Pathak, Manjusha Pandey, Siddharth Rautaray (2018) Applications of Deep Learning for object detection.
- [4] Sunil, Gagandeep (2019) Study of object detection methods and applications on digital images.
- [5] Zhong-Qiu Zhao, Shou-tao Xu, Xindong Wu (2017) Object detection with deep learning: A review.
- [6] Jun Wang Tingjuan Zhang, Yong Cheng, Najla Al-Nabhan (2021) Deep Learning for Object detection: A survey.
- [7] C.Stauffer and W.Grimson.Adaptive background mixture models for real-time tracking. Proceedings of IEEE Conf.Computer Vision and pattern Recognition,vol.2,1999, 246-252.
- [8] Girshick, R., Donahue, J., Darrel, T.,Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In: Computer Vision and Pattern Recognition. Columbus.2014, pp. 580-587.
- [9] Ren, S.Q., He, K.M., Girshick, R., Sun, J. Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. Montreal.2016, pp. 91-99.
- [10] Liu, W., Anguelov, D., Erhan, D., et al. SSD: Single Shot MultiBox Detector. European Conference on Computer Vision, 2016, pp. 21-37.
- [11] Dr. Sabeenian R.S, N Deivanai (2020) Wild animals intrusion detection using deep learning techniques.
- [12] Khyati Zalawadia ,Helly Shah ,Radhika Shringarure (2015) Traffic Detection and Diversion System: Case Study.