**A PROJECT REPORT**

**ON**

**OBJECT DETECTION USING MACHINE LEARNING**

**Abstraction**

In recent years, machine learning techniques have shown to perform well on a large variety of problems both in computer vision and Object detection. The rise of object detection is also in demand to use precise techniques in order to detect object efficiently. Hence, a techniques that fits in the class of efficiency is machine learning techniques. Here in our project we are pushing forward the concept of object detection as well as classification of objects using machine learning techniques.

The project is focused on trying to solve the problems associated with the issues of accuracy. We are also intended to enhance the attributes of our projects by bringing localization into the picture which will also tell us about the distance of the images.

Practically speaking this project is based on exhaustive comparison between various models and hence we are out with this project trying to offer something advanced and precise. We have used python as the programming language for coding. We have also introduced algorithms like YOLO, CNN, as well as RCNN.

We have also adopted the concepts of deep learning for high performance computing in our project.

CONTENTS

Chapter 1………………………....…………………………………………………………. 1

INTRODUCTION..............................................................................................….....1

Chapter 2 .....…………………………………………………………….………………….2

Literature review……………………………………………………….…………….3

Chapter 3 ……………………………………………………….………………………….…4

Software Requirements Specification.............................................................…...5

Chapter 4 …………………………………………………………….……………………….6

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)……………………….…...7

4.1 Stages of SDLC: …..……………………………….……………………………8

4.1.1 Planning and Requirement Analysis………………………………………..9

4.1.2 Defining Requirements ...........................…………………………………..8

4.1.3 Designing the Product Architecture…...........………………………………8

4.1.4 Building or Developing the Product………………………………..………8

4.1.5 Testing the Product ………………………………………………………...8

4.1.6 Deployment in the Market and Maintenance………………………….…....8

4.2 SDLC Model……………………………….……………………………………8

Chapter 5…………………………………………….……………………………………….8

Methodology…………………………………………….…………………..............8

Chapter 6……………………………………………………………….…………………….8

Data flow Diagram…………………………………………………………..………8

Chapter 7……………………………………………………………………….…………….8

7.1 Implementation……………………………………….…………………………8

7.2 Screenshot……………………………………………………….………………8

Chapter 8……………………………………………………………….…………………….8

System Testing…………………………………………………….………………...8

Chapter 9……………………………………………………….…………………………….8

Conclusion………………………………………….……………………………….8

REFERENCES………………………………………………………………………………8

LIST OF FIGURE

Figure 1 – Software Development Life Cycle………………..………………………………8

Figure 2 – Iterative Model- Design…...………………………………………………………8

Figure 3 – Object Detection………..…………………………………………………………8

Figure 4 – DFD Level 0………………………………………………………………………8

Figure 5 – DFD Level 1………………………………………………………………………8

Figure 6 – DFD Level 2………………………………………………………………………8

Figure 7 – Use Case Diagram…………………………………………………………………8

Figure 8 – OUTPUT….……………………………………………………………………….8

Chapter 1

Introduction

**1.1 Overview study:**

Efficient and accurate object detection has been an important topic in the advancement of computer vision systems. With the advent of machine learning technique the accuracy for object detection has increased significantly. The project aims to incorporate machine learning technique for detecting the object with the aim of providing high accuracy along with real time.

We present a novel approach to measure similarity and dissimilarity between shapes and exploit it for object recognition. In our system we are taking different parameters of an object and check it with the trained data set. Our system divides the input into an S\*S grid. Each grid will predict only one object. The object inside the grid box will be detected and recognized.

**1.2 Machine Learning:**

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. **Machine learning focuses on the development of computer programs** that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. **The primary aim is to allow the computers learn automatically** without human intervention or assistance and adjust actions accordingly.

**1.3 Deep Learning**

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised.

Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results

Chapter 2

**Literature review**

[1]Title: Shape Matching and Object Recognition Using Shape ContextsPublished in: IEEE Transactions on Pattern Analysis and Machine intelligence (Volume: 24 , Issue: 4 , Apr 2002 ) Author(s) : Belongie, Serge ; Malik, Jitendra ; Puzicha Jan **.**This paper present a novel approach to measuring similarity between shapes and exploit it for object recognition. In our framework, the measurement of similarity is preceded by 1) solving for correspondences between points on the two shapes, 2) using the correspondences to estimate an aligning transform. In order to solve the correspondence problem, we attach a descriptor, the shape context, to each point. The shape context at a reference point captures the distribution of the remaining points relative to it, thus offering a globally discriminative characterization. Corresponding points on two similar shapes will have similar shape contexts, enabling us to solve for correspondences as an optimal assignment problem.

[2]Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks Published in**:**2017 IEEE International Conference on Computer Vision (ICCV) author: Shaoqing Ren∗ Kaiming He Ross Girshick Jian Sun Microsoft Research {v-shren, kahe, rbg, jiansun}@microsoft.com.This paper present a conceptually simple, flexible, and general framework for object instance segmentation. Our approach efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance. The method, called Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps. Moreover, Mask R-CNN is easy to generalize to other tasks.

[3]You Only Look Once: Unified, Real-Time Object Detection Published in: *IEEE conference on computer vision & pattern recognition* 770 778, Year of Publication-2012, Author: Joseph Redmon University of Washington, Santosh Divvala Allen Institute for Artificial Intelligence

This paper present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance

[4]Real time object detection for smart vehicles Published in: Proceedings of the seventh IEEE International conference on computer vision INSPEC Accession Number**:**6371186, International Journal of Computer Applications, Volume 91 - Number 16 Year of Publication: 2014 Author: D.M GARVILA B.PHILOMIN.This paper presents an efficient shape-based object detection method based on Distance Transforms and describes its use for real-time vision on-board vehicles. The method uses a template hierarchy to capture the variety of object shapes; efficient hierarchies can be generated only for given shape distributions using stochastic optimization techniques (i.e. simulated annealing). Online, matching involves a simultaneous coarse-to-one approach over the shape hierarchy and over the transformation parameters. Very large speedup factors are typically obtained when comparing this approach with the equivalent brute-force formulation; we have measured gains of several orders of magnitudes.

[5] Real-time video-shot detection for scenesurveillance applications**Published in:**IEEE Transactions on Image Processing (Volume: 9, Issue: 1, Jan 2000) Author: E.Stringa, C.S. Regazzoni A surveillance system with automatic video-shot detection and indexing capabilities is presented. The proposed system aims at detecting the presence of abandoned objects in a guarded environment and at automatically performing online semantic video segmentation in order to facilitate the human operator's task of retrieving the cause of an alarm. The former task is performed by operating image segmentation based on temporal rank-order filtering, followed by classification in order to reduce false alarms. The latter task is performed by operating temporal video segmentation when an alarm is detected. In the clips of interest, the key frame is the one depicting a person leaving a dangerous object, and is determined on the basis of a feature indicating the movement around the dangerous region. Experimental results are reported in terms of static region detection, classification, clip and key-frame detection errors versus different levels of complexity of the guarded environment, in order to establish the performance that can be expected from the system in different situations.

Chapter 3

**SOFTWARE REQUIREMENTS SPECIFICATION**

A Software Requirements Specification (SRS) is a complete description of the behavior of the system to be developed. It includes the functional and non-functional requirements for the software to be developed. The functional requirement include what the software do and non-functional requirements include the constraint on the design or implementation of the system. Requirements must be measureable, testable, related to identified needs or opportunities, and defined to be a level of detail that is sufficient for system design.

**3.1 Overall Description**

The section provides a description of the general factors that affect the product, its requirements and also specifies the requirement, both for software and hardware which makes them easier to understand.

**3.1.1 Product Perspective**

The single shot detection system of detecting is to detect objects from the class of trained set with high accuracy and speed with higher fps. the key finding of the proposed system is the performance estimation in regards to trained data set using multiple object detection in a single shot.

**3.1.2 Product function**

The primary function of the system is to classify and localize object. The system has to do this and detect objects with relative high accuracy. The image frames that are provided are passed through the s detection system which takes upto 21 class, one additional if there is no object in the frame and check with the pre trained data set. The feature extractor and the detector network comes into play which checks the objects of variable sizes and then passing it through to non maximum suppression layer.

**3.1.4 Constraints**

The constraints of the proposed system are listed below:-

* The proposed system is good only for images with relatively larger size
* The proposed system works nicely only in proper lighting comditions
* Th system gets difficult to detect smaller objects

**3.1.5 Assumptions and Dependencies**

The system can detect objects that are there in the classes defined using the trained data set and the object should be of large enough size.

**3.2 Specific requirements**

This section of the SRS contains all the software requirements to a level of details required to enable the developer to design a system with those requirements and testers totest that the system satisfying those requirements.

**3.2.1 Software requirements**

The software requirements for implementing and testing of proposed system are-

* Operating system windows 7 or higher(64bit)

Techniques python 3.0 or higher,numpy,matploitlib

**3.2.2 Hardware requirements**

The hardware requirement specoication of proposed system are:-

* Processor Intel 64 bit processor
* Disk space 2GB
* RAM 2GB
* Input device Standard Keyboard and Mouse
* Output device VGA Camera and High resolution monitor

**3.2.3 Design constraints**

Design is one of the most important phases of software development. The constraint of this design is that the pre trained data set with the defined class of objects will work with the system.

**3.2.4 Interface**

An interface is the boundary across which two independent system meet and act together. For interface in this system we have designed it using tkinter with a snapshot button to take snapshot of the images.

Chapter 4

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC):

The software development life cycle (SDLC) is a framework defining tasks performed at each step in the software development process. SDLC is a structure followed by a development team within the software organization. The life cycle defines a methodology for improving the quality of software and the overall development process. The software development life cycle is also known as the software development process. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC.



**Figure 4.1 Software Development Life Cycle.**

**4.1 Stages of SDLC:**

A typical Software Development Life Cycle consists of the following stages –

**4.1.1 Planning and Requirement Analysis**

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales Department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

**4.1.2 Defining Requirements**

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an **SRS (Software Requirement Specification)** document which consists of all the product requirements to be designed and developed during the project life cycle.

**4.1.3 Designing the Product Architecture**

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS-Design Document Specification. The goal of this phase is to transform the requirements specified in the SRS document into a structure i.e. suitable for implementation in some programming.

**4.1.4 Building or Developing the Product**

In this stage of SDLC the actual development starts and the product is built. The Programming code is generated as per DDS during this stage. If the design is performed in a Mailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. The programming language is chosen with respect to the type of software being developed.

**4.1.5 Testing the Product**

This stage is usually a subset of all the stages as in the modem SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

**4.1.6 Deployment in the Market and Maintenance**

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT-User acceptance testing). Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

**4.2 SDLC Model**

4.2.1 Iterative Model- Design

Iterative process starts with a simple Implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

The following illustration is a representation of the Iterative and Incremental model -

Build 1

Implementation

Design & Development

Testing

Build 2

Implementation

Design & Development

Testing

Requirements

Build 3

Implementation

Design & Development

Testing

**Figure 4.2 Iterative Model-Design**

Iterative and Incremental development is a combination of both iterative design or iterative method and incremental build model for development. "During software development, more than one iteration of the software development cycle may be in progress at the same time.” This process may be described as an "evolutionary acquisition" or "incremental build approach”

4.2.1 Planning & Requirements: As with most any development project, the first step is go through an initial planning stage to map out the specification documents, establish software or hardware requirements, and generally prepare for the upcoming stages of the cycle.

4.2.2 Analysis & Design:Once planning is complete, an analysis is performed to nail down the appropriate business logic, database models, and the like that will be required at this stage in the project. The design stage also occurs here, establishing any technical requirements (languages, data layers, services, etc.) that will be utilized in order to meet the needs of the analysis stage.

4.2.3 Implementation: With the planning and analysis out of the way, the actual implementation and coding process can now begin. All planning, specification, and design docs up to this point are coded and implemented into this initial iteration of the project.

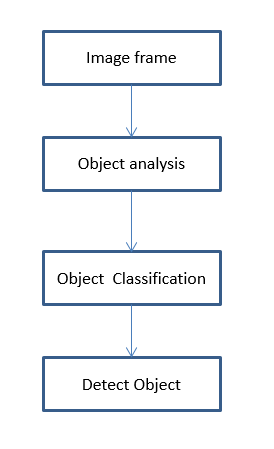
4.2.4 Testing: Once this current build iteration has been coded and implemented, the next step is to go through a series of testing procedures to identify and locate any potential bugs or issues that have cropped up.

Chapter 5

Methodology

**5.1 Methodology**

As we know that videos are made up of subsequent images (frames) which are move fast enough. So that, human eyes realize them continuous. Now, for any processing on video we need to the frame. The video analysis is done by following-



**Image frame:**

Image processing is any form of signal processing for which the input is an image, such as a digital photograph or video frame; the output of image processing may be either an image or a set of characteristics or parameters or features related to the image. Most image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal processing techniques to it. So Image Processing is essentially two-dimensional signal processing— this does involve quite a few new/modified techniques to deal with spatial information in 2D rather 1D data over time Image processing is any form of signal processing for which the input is an image, such as a digital photograph or video frame; the output of image processing may be either an image or a set of characteristics or parameters or features related to the image. Most image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal processing techniques to it. So Image Processing is essentially two-dimensional signal processing— this does involve quite a few new/modified techniques to deal with spatial information in 2D rather 1D data over time.

**Object analysis:**

Real Time Objectanalysis, as the name suggests, frames gets processed on the flow without any delay & loss of frames, as soon as it comes from image-sensor (camera).Basic criteria to perform real time image processing is:

Processing Time per Frame < Capturing Time per Frame.

Featuring Extraction & Detection, Pixel Manipulation or whatever needs to be done on every frame as soon as it’s arrives from camera sensor.

While trying to reduce processing time, take following precautions:

* Try to process byte-array coming from image sensor, instead of decompressing it in jpeg or any other format.
* Try to keep the frame resolution as small as possible because image processing is basically a Matrix Manipulation.
* Try to use grayscale images instead of RGB images, as it’s easy to process single channel image than going for 3 channel processing.

**Object classification:**

A method represents a class of objects by first acquiring a set of positive training images of the class of objects. A matrix A is constructed from the set of positive training images. Each row in the matrix A corresponds to a vector of intensities of pixels of one positive training image. Correlated intensities are grouped into a set of segments of a feature mask image. Each segment includes a set of pixels with correlated intensities. From each segment, a subset of representative pixels is selected. A set of features is assigned to each pixel in each subset of representative pixels of each segment of the feature mask image to represent the class of objects.

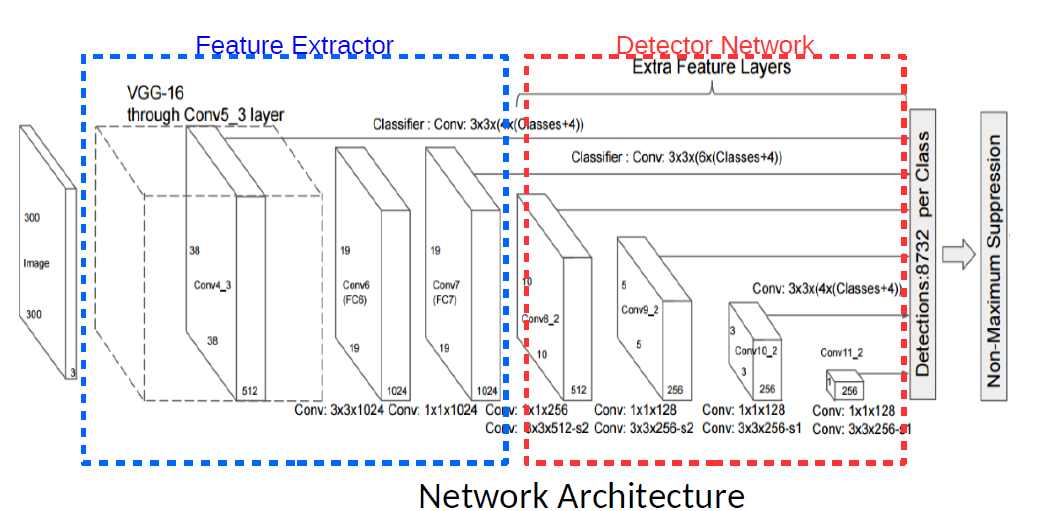
**Object Detection:**

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.

**Methodology approach:**

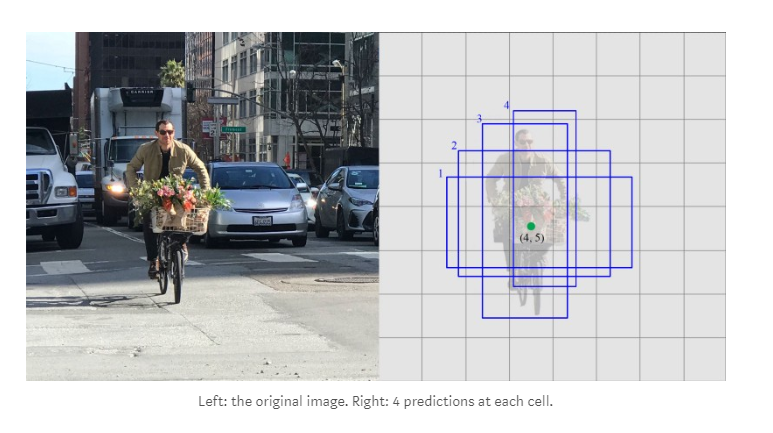
Single shot detection (SSD) Object detection composes of 2 parts:

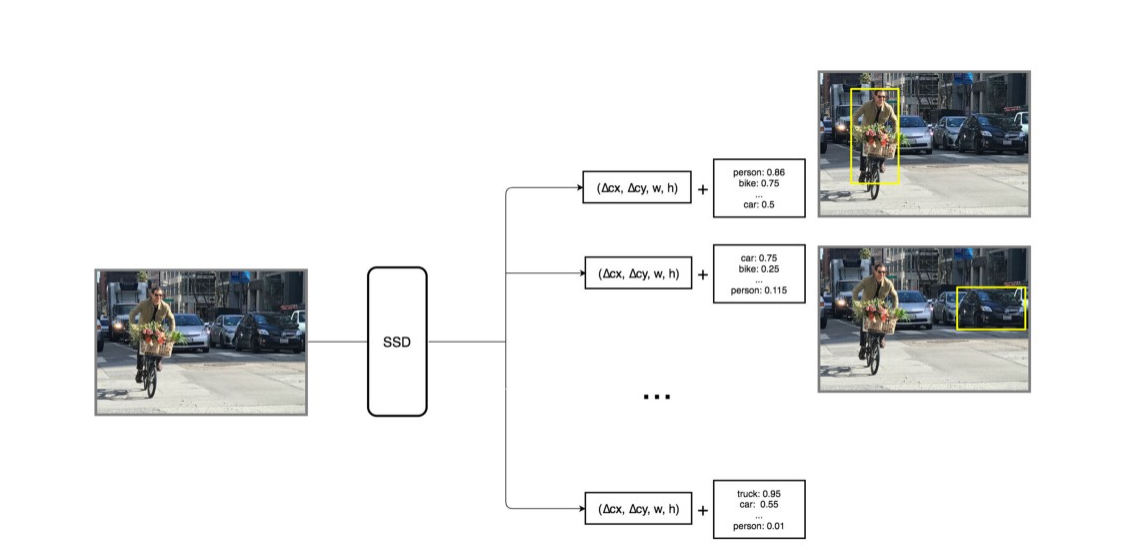
* Extract feature maps, and
* Apply convolution filters to detect objects



SSD uses **VGG16** to extract feature maps. Then it detects objects using the **Conv4\_3** layer.

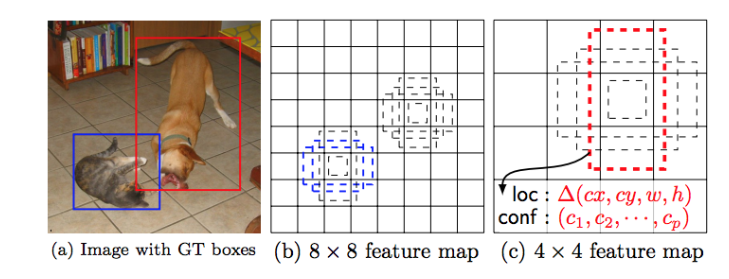
For illustration, we draw the Conv4\_3 to be 8 × 8 spatially (it should be 38 × 38). For each cell (also called location), it makes 4 object predictions.





Each prediction includes a boundary box and 21 scores for 21 classes

Multi-scale feature maps & default boundary boxes

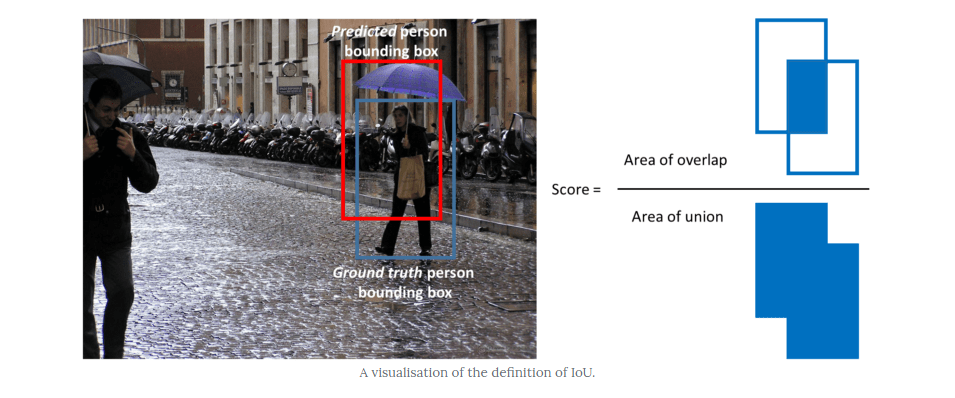


**Evaluation metric**

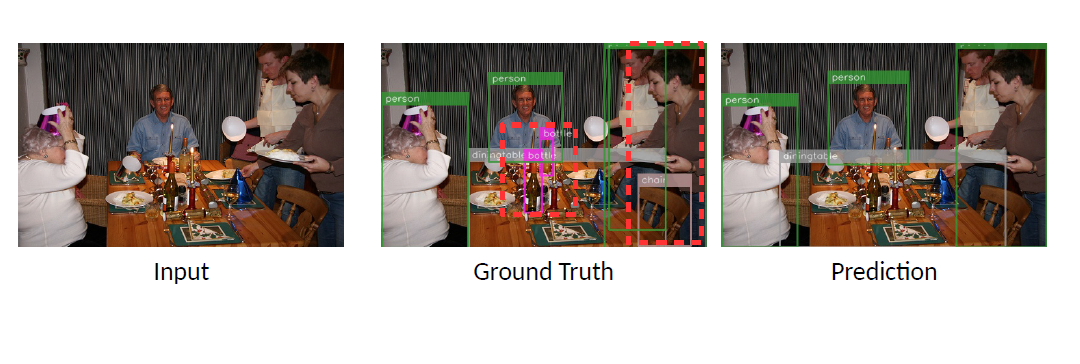
The most common evaluation metric that is used in object recognition tasks is ‘mAP’, which stands for ‘mean average precision’.

Each bounding box will have a score associated

Note that a detection is a true positive if it has an **‘intersection over union’** (IoU or overlap) with the ground-truth box greater than some threshold (usually 0.5). Instead of using mAP we typically use mAP@0.5 or mAP@0.25 to refer to the IoU that was used.



**Ground truth & prediction**



Chapter 6

**Data flow Diagram**

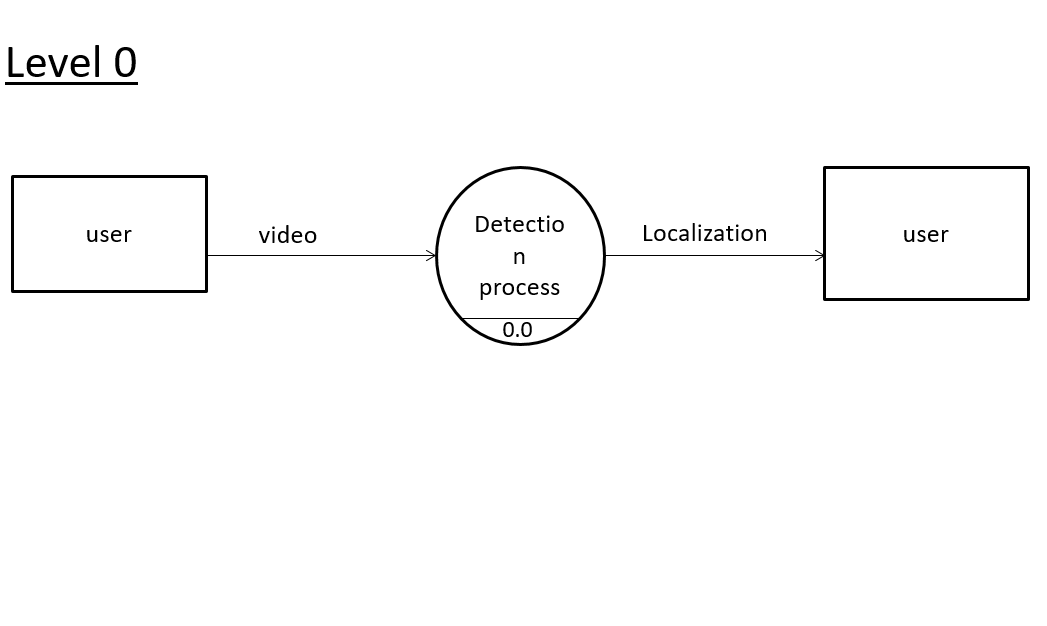
**Introduction**

A **data flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated.DFDs can also be used for the visualization of data processing (structured design).

**Symbols and Notations Used in DFDs**

1. **External entity ( ):** an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.
2. **Process ( ):**any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules. A short label is used to describe the process, such as “Submit payment.”
3. **Data store ( ):** files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as “Orders.”
4. **Data flow ( ):** the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labelled with a short data name, like “Billing details.”

**7.1 Level 0**

Figure 4 DFD Level 0

**7.2 Level 1**

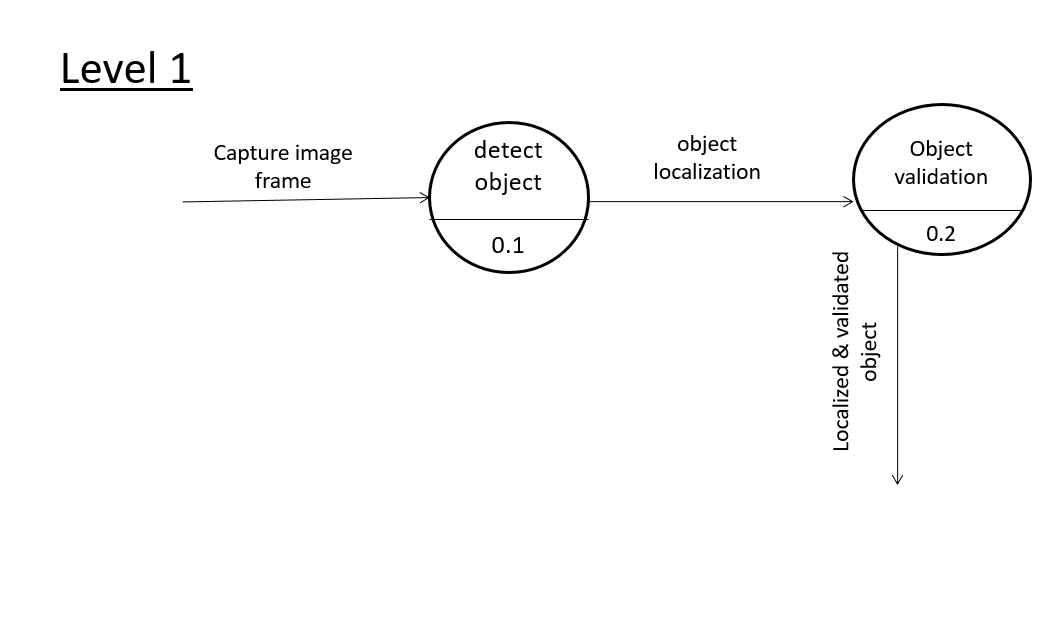


Figure 5 DFD Level 1

**7.3 Level 2**

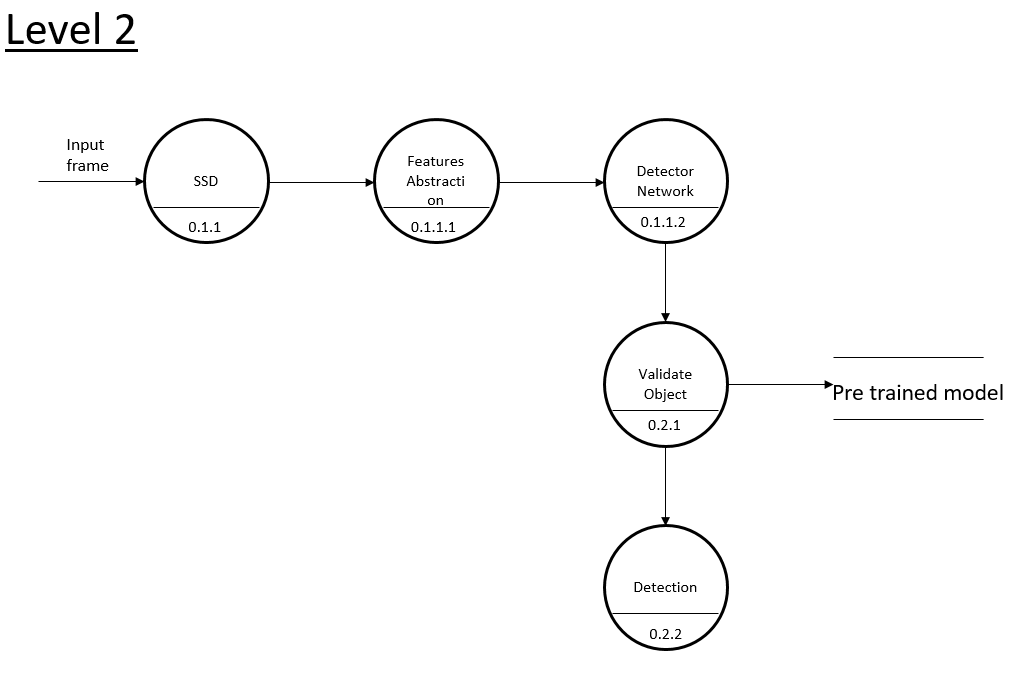
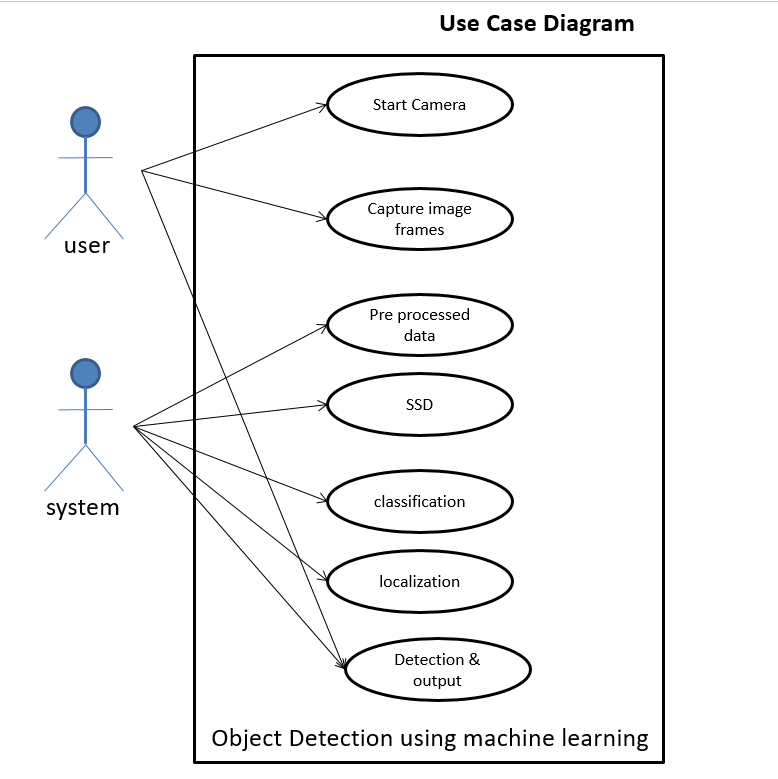


Figure 6 DFD Level 2

**Use Case Diagram**



Figures 7 Use case diagram

Chapter 7

Implementation

**7.1 Implementation**

# import the necessary packages

import tkinter

import PIL.Image, PIL.ImageTk

from fps import FPS

import numpy as np

import time

import cv2

from args import args

from classes import CLASSES

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

class App:

def \_\_init\_\_ (self, window, window\_title, video\_source=0):

self.window = window

self.window.resizable(0, 0)

self.window.title(window\_title)

self.video\_source = video\_source

# open video source (by default this will try to open the computer webcam)

self.vid = MyVideoCapture (self.video\_source)

# Create a canvas that can fit the above video source size

self.canvas = tkinter.Canvas (window, width=self.vid.width, height=self.vid.height)

self.canvas.pack ()

# Button that lets the user take a snapshot

self.btn\_snapshot = tkinter.Button (window, text="Snapshot", width=50, command=self.snapshot)

self.btn\_snapshot.pack (anchor=tkinter.CENTER, expand=True)

self.btn\_exit = tkinter.Button(window, text="Exit", width=50, command=self.exit\_window)

self.btn\_exit.pack (anchor=tkinter.CENTER, expand=True)

# After it is called once, the update method will be automatically called every delay milliseconds

self.delay = 15

self.update ()

self.window.mainloop ()

def exit\_window (self):

print("[INFO] closing...")

self.window.destroy ()

cv2.destroyAllWindows () # it is not mandatory in this application

def snapshot (self):

# Get a frame from the video source

ret, frame = self.vid.get\_frame ()

if ret:

cv2.imwrite ("snapshot/frame-" + time.strftime ("%d-%m-%Y-%H-%M-%S") + ".jpg", cv2.cvtColor (frame, cv2.COLOR\_RGB2BGR))

def update (self):

# Get a frame from the video source

ret, frame = self.vid.get\_frame ()

if ret:

self.photo = PIL.ImageTk.PhotoImage (image=PIL.Image.fromarray (frame))

self.canvas.create\_image (0, 0, image=self.photo, anchor=tkinter.NW)

self.window.after (self.delay, self.update)

class MyVideoCapture:

print ("[INFO] loading model...")

net = cv2.dnn.readNetFromCaffe (args["prototxt"], args["model"])

def \_\_init\_\_ (self, video\_source=0):

# Open the video source

print ("[INFO] starting video stream...")

self.vid = cv2.VideoCapture(video\_source)

time.sleep (2.0)

self.fps = FPS ().start ()

if not self.vid.isOpened ():

raise ValueError ("Unable to open video source", video\_source)

# Get video source width and height

self.width = self.vid.get (cv2.CAP\_PROP\_FRAME\_WIDTH)

self.height = self.vid.get (cv2.CAP\_PROP\_FRAME\_HEIGHT)

def get\_fram (self):

if self.vid.isOpened ():

ret, frame = self.vid. Read ()

(h, w) = frame. Shape [:2]

blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),

0.007843, (300, 300), 127.5)

self.net.setInput (blob)

detections = self.net.forward ()

for i in np.arange (0, detections.shape[2]):

confidence = detections [0, 0, i, 2]

if confidence > args["confidence"]:

idx = int (detections [0, 0, i, 1])

box = detections [0, 0, i, 3:7] \* np.array w, h, w, h])

(startX, startY, endX, endY) = box.astype ("int")

label = "{}: {:.2f}%".format(CLASSES[idx], confidence \* 100)

cv2.rectangle(frame, (startX, startY), (endX, endY), COLORS[idx], 2)

y = startY - 15 if startY - 15 > 15 else startY + 15

cv2.putText(frame, label, (startX, y), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, COLORS[idx], 2)

if ret:

# Return a boolean success flag and the current frame converted to BGR

return (ret, cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB))

else:

return (ret, None)

else:

return None

# Release the video source when the object is destroyed

def \_\_del\_\_ (self):

if self.vid.isOpened ():

self.vid.release()

self.fps.stop()

# Create a window and pass it to the Application object

App(tkinter.Tk(), "Object Detection App")

Chapter 8

System Testing

**8.1 Introduction**

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system’s compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware systems. The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together or between any of the assemblages and the hardware. System testing is a more limited type of testing, it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole. It is one of most important phase in testing process.

**8.2 Type of testing applied**

**8.2.1 White Box Testing**

White-box testing is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the expected outputs. White-box testing can be applied at the unit, integration and system levels of the software testing process. Although traditional testers tended to think of white-box testing as being done at the unit level, it is used for integration and system testing more frequently today. Though this method of test design can uncover many errors or problems, it has the potential to miss unimplemented parts of the specification or missing requirements.

**8.2.2 Black Box Testing**

Black-box Testing is a method of software testing that examines the functionality of an application without peering into its internal structure or working. This method of test can be applied virtually to every level of software testing: unit, integration, system and acceptance

**8.2.3 Unit Testing**

Unit testing, also known as Module Testing, focuses verification efforts on the module. The module is tested separately and this is carried out at the programming stage itself.

Unit test focuses on the smallest unit of software design- the software component or module.

**8.2.4 Performance Testing**

Performance testing determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization of the program unit. It occurs throughout all steps in the testing process.

**8.2.5 Integration Testing**

It is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with in the interface.

It takes the unit tested modules and builds a program structure.

**8.2.6 System Testing**

Tests to find the discrepancies between the system and its original objective, current

Specifications and system documentation.

Chapter 9

**Conclusion**

In this chapter. we presented an overview of the field of Object Detection, including historical developments, future potential, application areas, challenges, components and requirements, state-of-the-art systems, and UI concepts, SSD is a single-shot detector. It has no delegated region proposal network and predicts the boundary boxes and the classes directly from feature maps in one single pass.

To improve accuracy, SSD introduces:

* Small convolutional filters to predict object classes and offsets to default boundary boxes.
* Separate filters for default boxes to handle the difference in aspect ratios.
* Multi-scale feature maps for object detection.

SSD can be trained end-to-end for better accuracy. SSD makes more predictions and has a better coverage on location, scale and aspect ratios. With the improvements above, SSD can lower the input image resolution to 300 × 300 with a comparative accuracy performance. By removing the delegated region proposal and using lower resolution images, the model can run at real-time speed and still beats the accuracy of the state-of-the-art Faster R-CNN.

We have reached an important point in the progress toward object detection, in that the available technology is powerful enough for an increasing number of impressive research prototypes, but not yet sufficiently reliable, general, and comfortable for mass adoption. Compared to other applications described in this project, which are immediately realizable using today’s technology, it will take more time for object detection to reach the computing mainstream. However, object detection will have an enormous impact when it becomes commonplace. We are looking forward to further progress in the areas of computing visualisation technology, tracking accuracy and reliability, general scene understanding, and overall comfort. Finally, we hope that the benefits of object recognition will be achieved without compromising privacy and comfort Research and development in a field that could have a significant impact on social structures and conventions should be accompanied by careful consideration of how the commendable aspects of our social equilibrium can be protected and strengthened.

**References**

[1] Shape Matching and Object Recognition Using Shape Contexts Published in: IEEE Transactions on Pattern Analysis and Machine intelligence (Volume: 24, Issue: 4, Apr 2002)Author(s): Belongie, Serge; Malik, Jitendra; Puzicha Jan

[2] Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks published in: 2017 IEEE International Conference on Computer Vision (ICCV) author: Shaoqing Ren∗ Kaiming He Ross Girshick Jian Sun Microsoft Research {v-shren, kahe, rbg, jiansun}@microsoft.com

[3] You Only Look Once: Unified, Real-Time Object Detection Published in: *IEEE conference on computer vision & pattern recognition* 770 778, Year of Publication-2012, Author: Joseph Redmon University of Washington, Santosh Divvala Allen Institute for Artificial Intelligence

[4] Real time object detection for smart vehicles Published in: Proceedings of the seventh IEEE International conference on computer vision INSPEC Accession Number: 6371186, International Journal of Computer Applications, Volume 91 - Number 16 Year of Publication: 2014 Author: D.M GARVIL A B.PHILOMIN

[5] Real-time video-shot detection for scenesurveillance applications **Published in:**IEEE Transactions on Image Processing (Volume: 9, Issue: 1, Jan 2000) Author: E.Stringa, C.S. Regazzoni