OBJECT DETECTION AND IDENTIFICATION

A project report in partial fulfilment of the requirements for the

Award of degree of

Bachelor of Technology
in
Computer Science and Engineering

By

Ayush Maheshwari	**1629210027**
Divyanshi Agarwal	**1629210032**
Harsh Nain	**1629210038**
Kundan Jha	**1629210050**

Under the supervision of
Ayush Singhal
(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MEERUT INSTITUTE OF TECHNOLOGY, MEERUT

Affiliated to

DR. A.P.J ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW

Certificate

I hereby declare that the work which is being presented in the project report entitled, **"Object Detection and Identification"**, in partial fulfilment of the requirements for the award of degree of Bachelor of Technology submitted in Computer Science and Engineering of Meerut Institute of Technology, Meerut, is an authentic record of my own work carried out under the supervision of **Ayush Singhal** and refers to other researcher's works which are duly listed in the reference section. The matter presented in this Project has not been submitted for the award of any other degree of this or any other university.

The matter presented in this Project has not been submitted for the award of any other degree of this or any other university.

Name of Candidate

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Ayush Singhal
(Assistant Professor)
Meerut Institute of Technology,
MEERUT.

K.N. Tripathi
(Assistant Professor)
Meerut Institute of Technology,
MEERUT.

Countersigned by

Suraj Malik
HOD-CSE
Meerut Institute of Technology,
MEERUT

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe a special debt of gratitude to Mr. **Ayush Singhal** and Mr. **K.N. Tripathi**, Department of Computer Science & Engineering, Meerut Institute of Technology, Meerut for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavours have seen light of the day.

We also take the opportunity to acknowledge the contribution of Mr. **Suraj Malik**, Head, Department of Computer Science & Engineering, Meerut Institute of Technology, Meerut for his full support and assistance during the development of the project.

We also do not like to miss the opportunity to acknowledge the contribution of all project co-coordinators and faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Signature:
Name: *Ayush Maheshwari*
Roll No.:*1629210027*
Date:

Signature:
Name: *Divyanshi Agarwal*
Roll No.:*1629210032*
Date:

Signature:
Name: *Harsh Nain*
Roll No.:*1629210038*
Date:

Signature:
Name: *Kundan Jha*
Roll No.:1629210050
Date:

Abstract

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

LIST OF FIGURES

Chapter 1

Chapter 3

Chapter 4

Chapter 5

Chapter 6

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

Many problems in computer vision were saturating on their accuracy before a decade. However, with the rise of deep learning techniques, the accuracy of these problems drastically improved. One of the major problems was that of image classification, which is defined as predicting the class of the image. A slightly complicated problem is that of image localization, where the image contains a single object and the system should predict the class of the location of the object in the image (a bounding box around the object). The more complicated problem of object detection involves both classification and localization. In this case, the input to the system will be an image, and the output will be a bounding box corresponding to all the objects in the image, along with the class of objects in each box. Overview of this is in Figure 1.1.

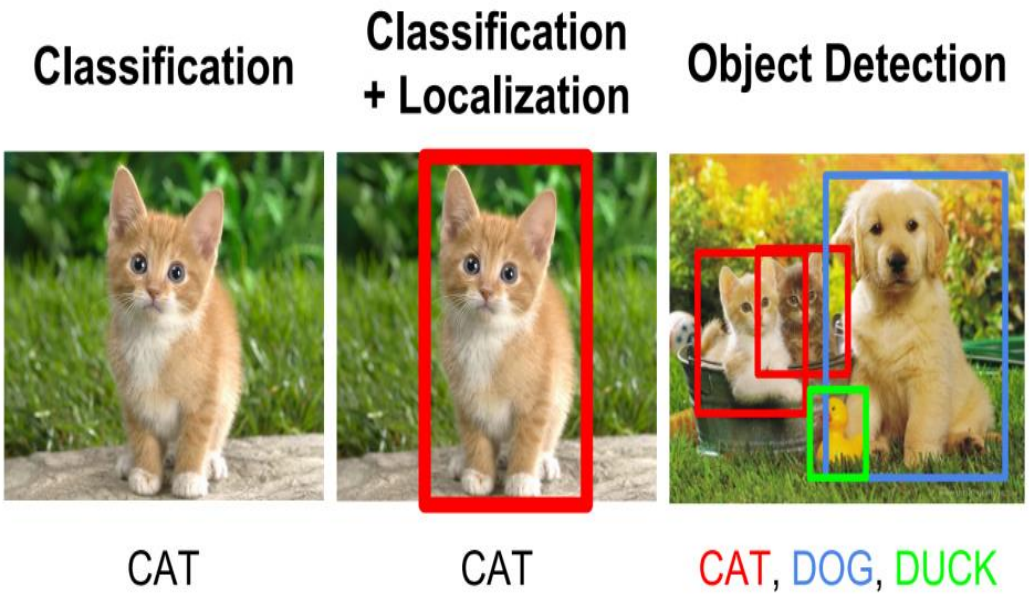


Figure 1.1: Object Detection Detecting Animals In The Image

1.2 Background of the problem

The aim of object detection is to detect all instances of objects from a known class, such as people, cars or faces in an image. Generally, only a small number of instances of the object are present in the image, but there is a very large number of possible locations and scales at which they can occur and that need to somehow be explored. Each detection of the image is reported with some form of pose information. This is as simple as the location of the object, a location and scale, or the extent of the object defined in terms of a bounding box. In some other situations, the pose information is more detailed and contains the parameters of a linear or non-linear transformation. For example, face detection in a face detector may compute the locations of the eyes, nose and mouth, in addition to the bounding box of the face. An example of a bicycle detection in an image that specifies the locations of certain parts is shown in Figure 1.1. The pose can also be defined by a three-dimensional transformation specifying the location of the object relative to the camera. Object detection systems always construct a model for an object class from a set of training examples. In the case of a fixed rigid object in an image, only one example may be needed, but more generally multiple training examples are necessary to capture certain aspects of class variability.

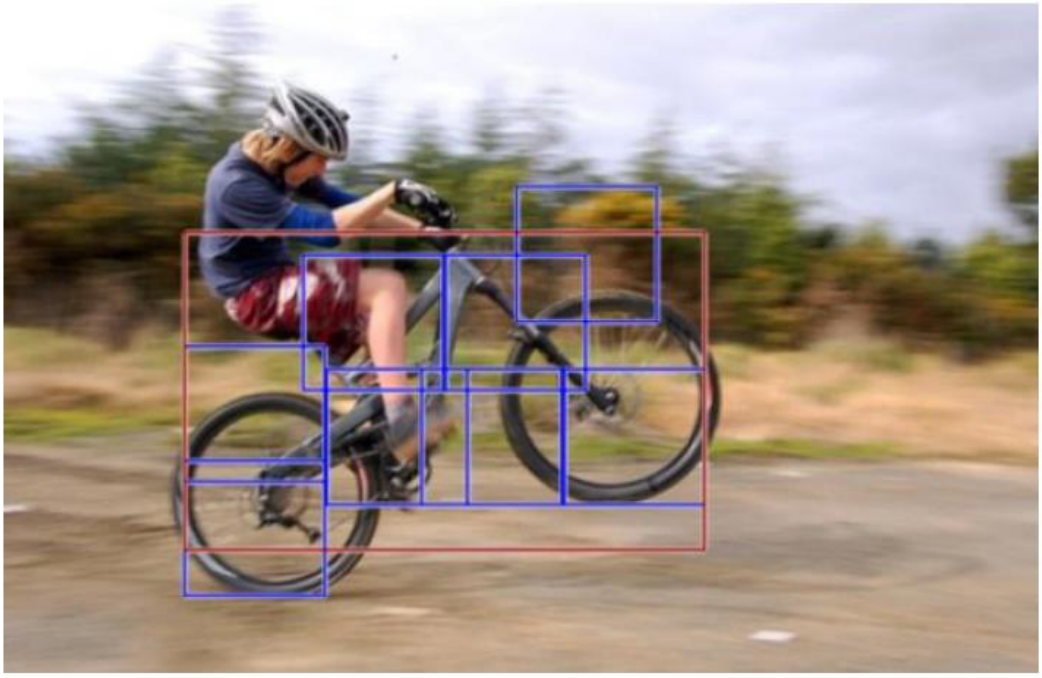


Figure 1.2: Object Detection Detecting Animals In The Image

Convolution implementation of the sliding windows Before we discuss the implementation of the sliding window using convents, let us analyse how we can convert the fully connected layers of the network into convolutional layers. Figure 1.2 shows a simple convolutional network with two fully connected layers each of shape.

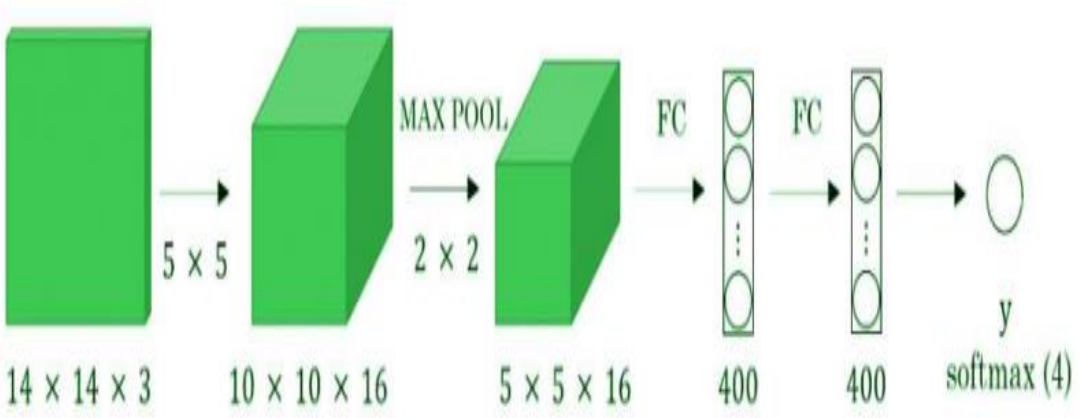


Figure 1.3: Simple Convolution Network

A fully connected layer can be converted to a convolutional layer with the help of a 1D convolutional layer. The width and height of this layer is equal to one and the number of filters are equal to the shape of the fully connected layer. An example of this is shown in Figure 1.3.

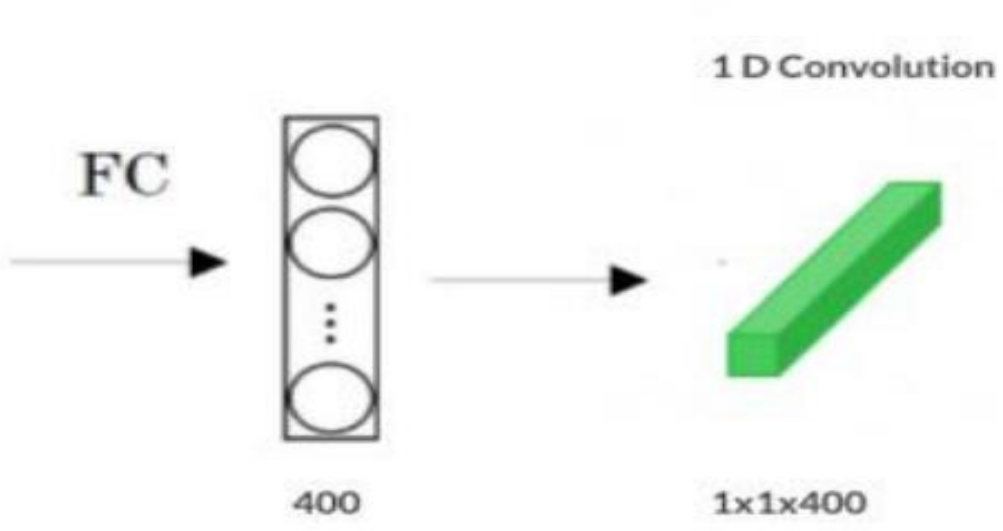


Figure 1.4: Conversion of 1d Fully Connected to Convolutional Layer

We can apply the concept of conversion of a fully connected layer into a convolutional layer to the model by replacing the fully connected layer with a 1D convolutional layer. The number of filters of the 1D convolutional layer is equal to the shape of the fully connected layer. This representation is shown

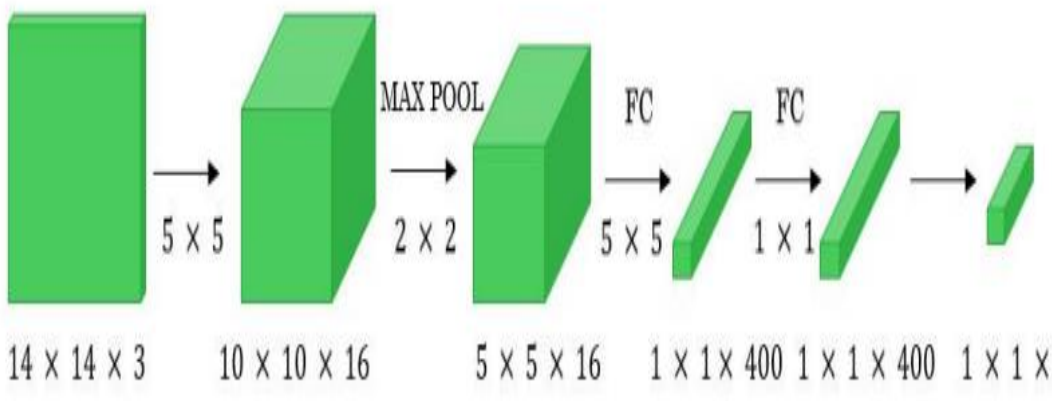


Figure 1.5: Conversion of 1d Fully Connected To Convolutional Layer

in Figure 1.5. Also, the output SoftMax layer is also a convolutional layer of shape (1, 1, 4), where 4 is the number of classes to predict.

Now, let's extend the above approach to implement a convolutional version of the sliding window. First, let us consider the ConvNet that we have trained to be in the following representation (no fully connected layers).

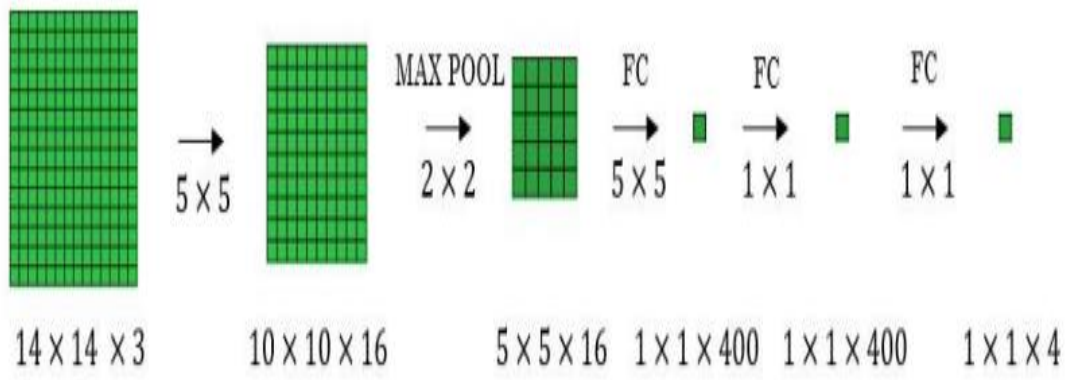
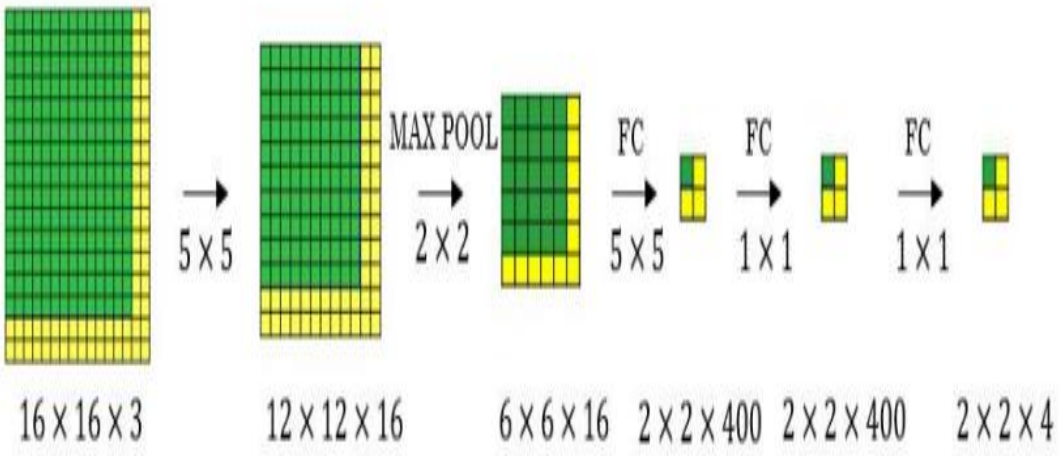


Figure 1.6: Convolutional Version of Sliding Window

Let's assume the size of the input image to be 16 × 16 × 3. If we are using the sliding window approach, then we would have passed this image to the ConvNet four times, where each time the sliding window crops the part of the input image matrix of size 14 × 14 × 3 and pass it through the ConvNet. But instead of this, we feed the full image (with shape 16 × 16 × 3) directly into the trained ConvNet (see Figure 1.7). These results will give an output matrix of shape 2 × 2 × 4. Each cell in the output matrix represents the result of the possible crop and the classified value of the cropped image. For example, the left cell of the output matrix(the green one) in Figure1.7 represents the



result of the first sliding window. The other cells in the matrix represent the results of the remaining sliding window operations.

Figure 1.7: Result Of First Sliding Window

The stride of the sliding window is decided by the number of filters used in the Max Pool layer. In the example above, the Max Pool layer has two filters, and for the result, the sliding window moves with a stride of two resulting in four possible outputs to the given input. The main advantage of using this technique is that the sliding window runs and computes all values simultaneously. Consequently, this technique is really fast. The weakness of this technique is that the position of the bounding boxes is not very accurate.

A better algorithm that tackles the issue of predicting accurate bounding boxes while using the convolutional sliding window technique is the YOLO algorithm. YOLO stands for you only look once. It is popular because it achieves high accuracy while running in real-time. This algorithm requires only one forward propagation pass through the network to make the predictions. This algorithm divides the image into grids and then runs the image classification and localization algorithm (discussed under object localization) on each of the grid cells. For example, we can give an input image of size 256 × 256. We place a 3 × 3 grid on the image (see Figure 1.8)

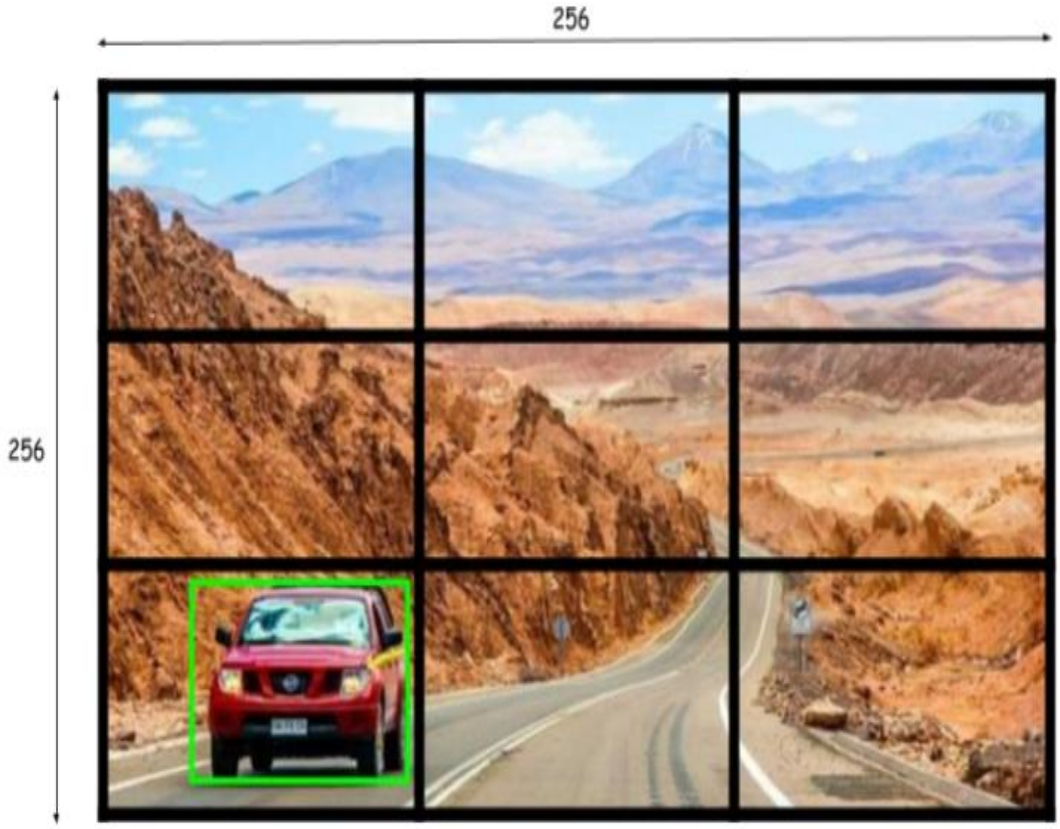


Figure 1.8: Divide The Input Image(Size Is 256 X 256) Into Grid Cell

Next, we shall apply the image classification and localization algorithm on each grid cell.In the image of each grid cell, the target variable is defined as Yi,j=[pcbxbybhbwc1c2c3c4]T(6) Do everything once with the convolution sliding

window. Since the shape of the target variable for each grid cell in the image is 1 × 9 and there are 9 (3 × 3) grid cells, the final output of the model will be:

$$Final\ Output = \underbrace{3 \times 3}_{\text{Number of grid cells}} \times \underbrace{9}_{\text{Output label for each grid cell}}$$

The advantages of the YOLO algorithm is that it is very fast and predicts much more accurate bounding boxes. Also, in practice to get the more accurate predictions, we use a much finer grid, say 19 × 19, in which case the target output is of the shape 19 × 19 × 9.

CHAPTER 2

SOFTWARE REQUIREMENT SPECIFICATION

2.1 Hardware Specification

The CPU must be of i5 2015 MBP (frames per second minimum to be 5) and quad-core ARM Cortex-A57 CPU. The GPU must be of the Nvidia TitanXP (frames per second greater than 70). It was selected because of its low power consumption which is estimated lower than 20W when fully utilized, yet its high computational capacity per watt. Moreover, this system-on-a-chip supports standard and widely used frameworks such as Compute Unified Device Architecture (CUDA). In addition, it supports 16-bit floating point operations which may enable to reach a better performance if precision is not the most important aspect.

2.2 Software Specification

(i) Python 3:

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

(i) TensorFlow:

TensorFlow is an open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks, etc. It is for internal Google use and released under the Apache License 2.0 on November 9, 2015.

TensorFlow can run on multiple CPU’s and is available on various platforms such as 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors.

Command - ***pip install TensorFlow***

(ii) NumPy:

NumPy is a library of Python programming language, adding support for large, multidimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate over these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several developers. In 2005 Travis Oliphant created NumPy by incorporating features of computing Numarray into Numeric, with extension modifications. NumPy is open-source software and has many contributors.

Command - ***pip install numpy***

(iii) SciPy:

SciPy contains modules for many optimizations, linear algebra, integration, interpolation, special function, FFT, signal and image processing, ODE solvers and other tasks common in engineering.

SciPy abstracts majorly on NumPy array objects, and is the part of the NumPy stack which includes tools like Matplotlib, pandas and SymPy etc., and an expanding set of scientific computing libraries. This NumPy stack has similar uses to other applications such as MATLAB, Octave, and Scilab. The NumPy stack is also sometimes referred to as the SciPy stack.

Command - pip install scipy

(iv) OpenCV:

OpenCV is a library of programming functions mainly aimed at real time computer vision. originally developed by Intel, it is later supported by Willow Garage then Itseez. The library is cross-platform and free to use under the open-source BSD license.

Command - pip install opencv-python

(v) Pillow:

Python Imaging Library is a free Python programming language library that provides support to open, edit and save several different formats of image files. Windows, Mac OS X and Linux are available for this.

Command - pip install pillow

(vi) Matplotlib:

Matplotlib is a Python programming language plotting library and its NumPy numerical math extension. It provides an object-oriented API to use general-purpose GUI toolkits such as Tkinter, wxPython, Qt, or GTK+ to embed plots into applications.

Command - pip install matplotlib

(vii) H5py:

The software h5py includes a high-level and low-level interface for Python's HDF5 library. The low interface is expected to be complete wrapping of the HDF5 API, while the high-level component uses established Python and NumPy concepts to support access to HDF5 files, datasets and groups.

A strong emphasis on automatic conversion between Python (Numpy) data types and data structures and their HDF5 equivalents vastly simplifies the process of reading and writing data from Python.

Command - pip install h5py

(viii) Keras:

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

Command - pip install keras

(ix) ImageAI:

ImageAI provides an API to recognize 1000 different objects in a picture using pre-trained models that were trained on the ImageNet-1000 dataset. The model implementations provided are SqueezeNet, ResNet, InceptionV3 and DenseNet.

Command - pip3 install imageai --upgrade

CHAPTER 3

FEASIBILITY STUDY

3.1 Viability of Object detection

The motive of object detection is to recognize and locate (localize) all known objects in a scene.

Preferably in 3D space, recovering the pose of objects in 3D is very important for robotic control systems.

The information from the object detector can be used for obstacle avoidance and other interactions with the environment.

One of the best examples of why you need object detection is the high-level algorithm for autonomous driving:

- In order for a car to decide what to do next: accelerate, apply brakes or turn, it needs to know where all the objects are around the car and what those objects are
- That requires object detection
- You would essentially train the car to detect known set of objects: cars, pedestrians, traffic lights, road signs, bicycles, motorcycles, etc.

3.2 Challenges for object detection

3.2.1 Dual priorities: object classification and localization

The first major complication of object detection is its added goal: not only do we want to classify image objects but also to determine the objects' positions, generally referred to as the *object localization* task. To address this issue, researchers most often use a multi-task loss function to penalize both misclassifications and localization errors.

Regional-based CNNs represent one popular class of object detection frameworks. These methods consist of the generation of region proposals where objects are likely to be located followed by CNN processing to classify and further refine object locations. Ross Girshick et al. developed **Fast R-CNN** to improve upon their initial results with **R-CNN**. As its name implies, Fast R-CNN provides a dramatic speed-up, but accuracy also improves because the classification and localization tasks are optimized

using one unified multi-task loss function. Each candidate region that may contain an object is compared to the image's true objects. Candidate regions then incur penalties for both false classifications and misalignment of the bounding boxes. Hence, the loss function consists of two kinds of terms:

$$\mathcal{L}(p, u, t^u, v) = \overbrace{\mathcal{L}_c(p, u)}^{classification} + \lambda \overbrace{[u \geq 1] \mathcal{L}_l(t^u, v)}^{localization},$$

where the classification term imposes log loss on the predicted probability of the true object class u and the localization term is a smooth L_1 loss for the four positional components that define the rectangle. Note that the localization penalty does not apply to the background class when no object is present, $u=0$. Also note that the parameter λ may be adjusted to prioritize either classification or localization more strongly.

3.2.2 Speed for real-time detection

Object detection algorithms need to not only accurately classify and localize important objects, they also need to be incredibly fast at prediction time to meet the real-time demands of video processing. Several key enhancements over the years have boosted the speed of these algorithms, improving test time from the 0.02 frames per second (fps) of R-CNN to the impressive 155 fps of Fast YOLO.

Fast R-CNN and **Faster R-CNN** aim to speed up the original R-CNN approach. R-CNN uses **selective search** to generate 2,000 candidate regions of interest (RoIs) and passes each RoI through a CNN base individually, which causes a massive bottleneck since the CNN processing is quite slow. Fast R-CNN instead sends the entire image through the CNN base just once and then matches the RoIs created with selective search to the CNN feature map, yielding a 20-fold reduction in processing time. While Fast R-CNN is much speedier than R-CNN, yet another speed barrier persists. It takes approximately 2.3 seconds for Fast R-CNN to perform object detection on a single image, and selective search accounts for a full 2 seconds of that time! Faster R-CNN

replaces selective search with a separate sub-neural network to generate RoIs, creating another 10x speed up and thus testing at a rate of about 7–18 fps.

Despite these impressive improvements, videos are typically shot at at least 24 fps, meaning Faster R-CNN will likely not keep pace. Regional-based methods consist of two separate phases: proposing regions and processing them. This task separation proves to be somewhat inefficient. Another major type of object detection system relies on a unified one-state approach instead. These so-called single-shot detectors fully locate and classify objects during a single pass over the image, which substantially decreases test time. One such single-shot detector, YOLO begins by laying out a grid over the image and allows each grid cell to detect a fixed number of objects of varying sizes. For each true object present in the image, the grid cell associated with the object's center is responsible for predicting this object. A complex, multi-term loss function then ensures that all localization and classification occurs within one process. One version of this method, Fast YOLO, has even achieved rates of 155 fps; however, classification and localization accuracy drops off sharply at this elevated speed.

Ultimately, today's object detection algorithms attempt to strike a balance between speed and accuracy. Several design choices beyond the detection framework influence these outcomes. For example, **YOLOv3** allows images of varying resolution: high-res images typically see better accuracy but slower processing times and vice versa for low-res images. The choice of the CNN base also affects the speed-accuracy tradeoff. Very deep networks like the 164 layers used in Inception-ResNet-V2 yield impressive accuracy, but pale in comparison to frameworks with VGG-16 in terms of speed. Object detection design choices must be made in context depending on whether speed or accuracy takes priority.

3.2.3 Multiple spatial scales and aspect ratios

For many applications of object detection, items of interest may appear in a wide range of sizes and aspect ratios. Practitioners leverage several techniques to ensure detection algorithms are able to capture objects at multiple scales and views.

I. Anchor boxes

Instead of selective search, Faster R-CNN's updated region proposal network uses a small sliding window across the image's convolutional feature map to generate candidate RoIs. Multiple RoIs may be predicted at each position and are described relative to reference *anchor boxes*. The shapes and sizes of these anchor boxes are carefully chosen to span a range of different scales and aspect ratios. This allows various types of objects to be detected with the hopes that the bounding box coordinates need not be adjusted much during the localization task. Other frameworks, including single-shot detectors, also adopt anchor boxes to initialize regions of interest.

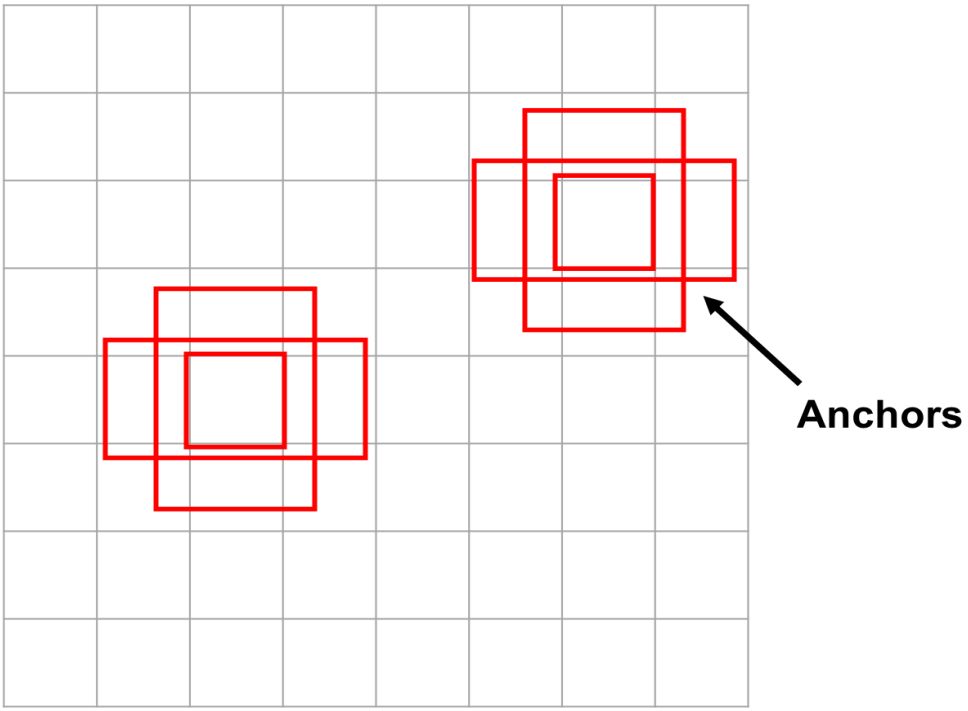


Figure 3.1: Carefully Chosen Anchor Boxes Of Varying Sizes And Aspect Ratio

II. Multiple feature maps

Single-shot detectors must place special emphasis on the issue of multiple scales because they detect objects with a single pass through the CNN framework. If objects are detected from the final CNN layers alone, only large items will be found as smaller

items may lose too much signal during down sampling in the pooling layers. To address this problem, single-shot detectors typically look for objects within multiple CNN layers including earlier layers where higher resolution remains. Despite the precaution of using multiple feature maps, single-shot detectors notoriously struggle to detect small objects, especially those in tight groupings like a flock of birds.

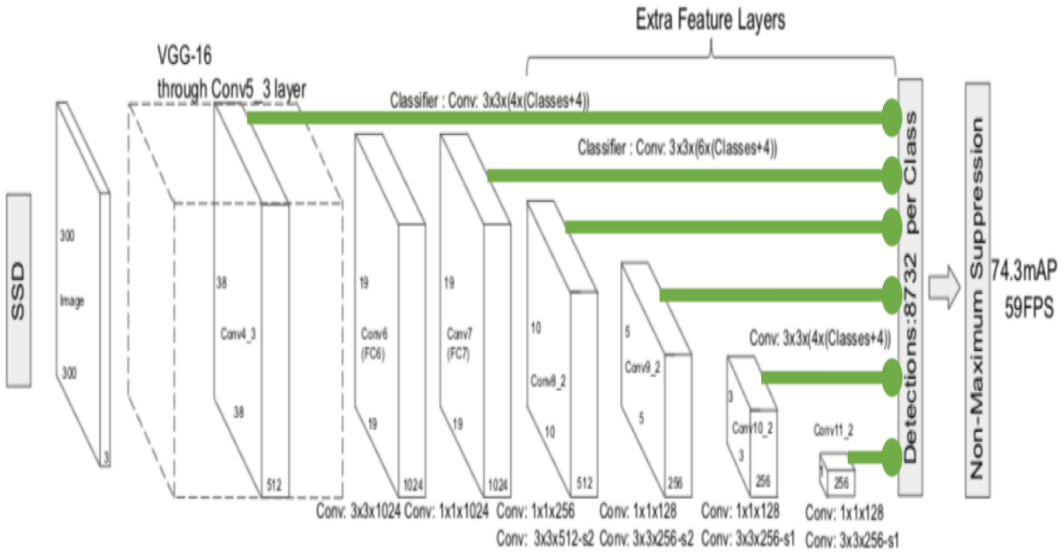


Figure 3.2: Feature Maps from Multiple CNN Layers Help Predict Objects At Multiple Scales

III. Feature pyramid network

The **feature pyramid network (FPN)** takes the concept of multiple feature maps one step further. Images first pass through the typical CNN pathway, yielding semantically rich final layers. Then to regain better resolution, FPN creates a top-down pathway by up sampling this feature map. While the top-down pathway helps detect objects of varying sizes, spatial positions may be skewed. Lateral connections are added between the original feature maps and the corresponding reconstructed layers to improve object localization. FPN currently provides one of the leading ways to detect objects at multiple scales, and YOLO was augmented with this technique in **its 3rd version**.

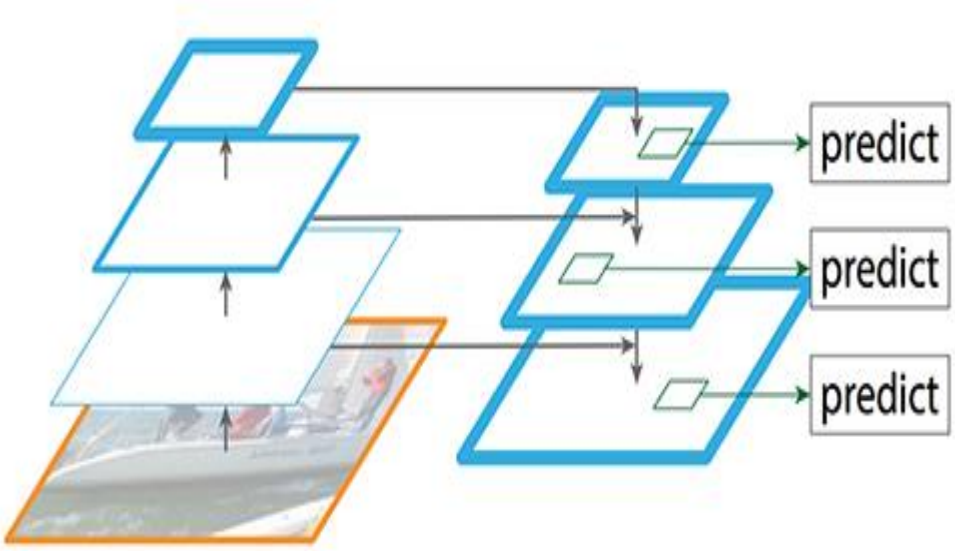


Figure 3.3: Feature Pyramid Network Detects Objects Of Varying Sizes By Reconstructing High-resolution Layers From Layers With Greater Semantic Strength

3.2.4 Limited data

The limited amount of annotated data currently available for object detection proves to be another substantial hurdle. Object detection datasets typically contain ground truth examples for about a dozen to a hundred classes of objects, while image classification datasets can include upwards of 100,000 classes. Furthermore, crowdsourcing often produces image classification tags for free (for example, by parsing the text of user-provided photo captions). Gathering ground truth labels *along with* accurate bounding boxes for object detection, however, remains incredibly tedious work.

The COCO dataset, provided by Microsoft, currently leads as some of the best object detection data available. COCO contains 300,000 segmented images with **80 different categories** of objects with very precise location labels. Each image contains about 7 objects on average, and items appear at very broad scales. As helpful as this dataset is, object types outside of these 80 select classes will not be recognized if training solely on COCO.

A very interesting approach to alleviating data scarcity comes from YOLO9000, the **second version of YOLO**. YOLO9000 incorporates many important updates into YOLO, but it also aims to narrow the dataset gap between object detection and image

classification. YOLO9000 trains simultaneously on both COCO and **ImageNet**, an image classification dataset with tens of thousands of object classes. COCO information helps precisely locate objects, while ImageNet increases YOLO's classification "vocabulary." A hierarchical Word Tree allows YOLO9000 to first detect an object's concept (such as "animal/dog") and to then drill down into specifics (such as "Siberian husky"). This approach appears to work well for concepts known to COCO like animals but performs poorly on less prevalent concepts since RoI suggestion comes solely from the training with COCO.

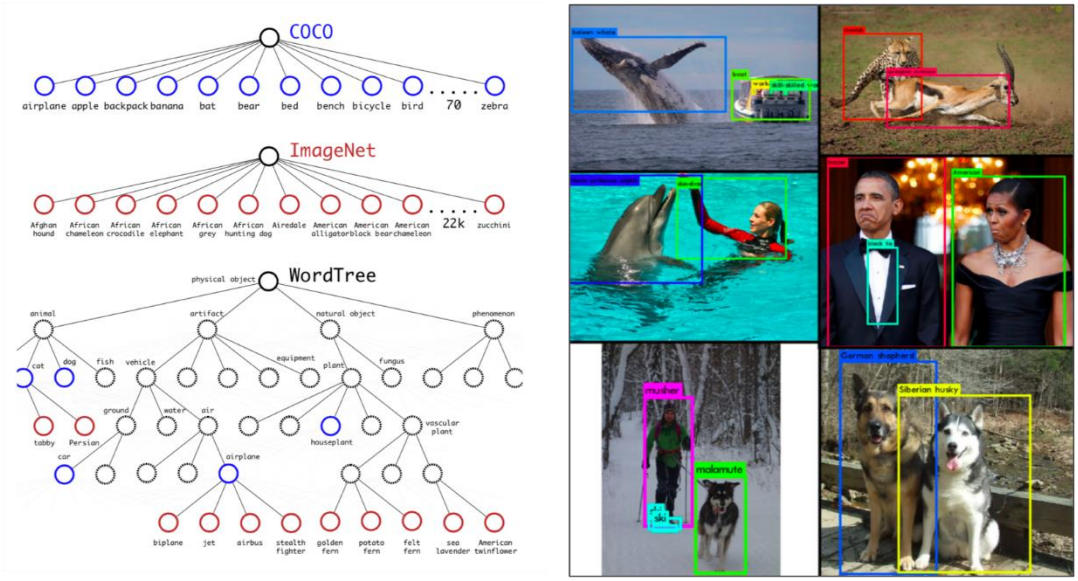


Figure 3.4: Yolo9000 Trains on Both Coco and ImageNet To Increase Classification "Vocabulary"

3.2.5 Class imbalance

Class imbalance proves to be an issue for most classification problems, and object detection is no exception. Consider a typical photograph. More likely than not, the photograph contains a few main objects and the remainder of the image is filled with background. Recall that selective search in R-CNN produces 2,000 candidate RoIs per image–just imagine how many of these regions do not contain objects and are considered negatives!

A recent approach called focal loss is implemented in **RetinaNet** and helps diminish the impact of class imbalance. In the optimization loss function, focal loss replaces the traditional log loss when penalizing misclassifications:

$$FL(p_u) = -\overbrace{(1-p_u)^{\gamma}}^{*} \log(p_u)$$

where p_u is the predicted class probability for the true class and $\gamma > 0$. The additional factor (*) reduces loss for well-classified examples with high probabilities, and the overall effect de-emphasizes classes with many examples that the model knows well, such as the background class. Objects of interest occupying minority classes, therefore, receive more significance and see improved accuracy.

3.3 Overall view for object detection

Object detection is customarily considered to be much harder than image classification, particularly because of these five challenges: dual priorities, speed, multiple scales, limited data, and class imbalance. Researchers have dedicated much effort to overcome these difficulties, yielding oftentimes amazing results; however, significant challenges still persist.

Basically, all object detection frameworks continue to struggle with small objects, especially those bunched together with partial occlusions. Real-time detection with top-level classification and localization accuracy remains challenging, and practitioners must often prioritize one or the other when making design decisions. Video tracking may see improvements in the future if some continuity between frames is assumed rather than processing each frame individually. Furthermore, an interesting enhancement that may see more exploration would extend the current two-dimensional bounding boxes into three-dimensional bounding cubes. Even though many object

detection obstacles have seen creative solutions, these additional considerations–and plenty more–signal that object detection research is certainly not done!

3.4 Technical Feasibility Study

3.4.1 Hardware Requirements

The algorithms that are shown throughout this paper are designed for a specific target, NVIDIA Jetson TX1. This system-on-a-chip contains a CPU module and a GPU module. The CPU module is a quad-core ARM Cortex-A57 CPU. The GPU module is a Maxwell GPU that has 256 cores, evenly distributed onto two Stream Multiprocessors. It was selected because of its low power consumption which is estimated lower than 20W when fully utilized, yet its high computational capacity per watt. Moreover, this system-on-a-chip supports standard and widely used frameworks such as Compute Unified Device Architecture (CUDA). In addition, it supports 16-bit floating point operations which may enable to reach a better performance if precision is not the most important aspect.

3.4.2 Software Requirement

Install Python on your computer system

1. Install ImageAI and its dependencies like tensorflow, Numpy, OpenCV, etc.

2. Download the Object Detection model file (yolo)

Steps to be followed: -

1) Download and install Python version 3 from official Python Language website

https://python.org

2) Install the following dependencies via pip:

(i) Tensorflow:

Tensorflow is an open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks, etc. It is for internal Google use and released under the Apache License 2.0 on November 9, 2015.

Tensorflow can run on multiple CPU's and is available on various platforms such as 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Tensorflow computations are expressed as stateful dataflow graphs. The name Tensorflow derives from operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors.

Command - pip install tensorflow

(ii) Numpy:

NumPy is a library of Python programming language, adding support for large, multidimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate over these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several developers. In 2005 Travis Olphant created NumPy by incorporating features of computing Numarray into Numeric, with extension modifications. NumPy is open-source software and has many contributors.

Command - pip install numpy

(iii) SciPy:

SciPy contains modules for many optimizations, linear algebra, integration, interpolation, special function, FFT, signal and image processing, ODE solvers and other tasks common in engineering.

SciPy abstracts majorly on NumPy array objects, and is the part of the NumPy stack which includes tools like Matplotlib, pandas and SymPy etc., and an expanding set of scientific computing libraries. This NumPy stack has similar uses to other applications such as MATLAB, Octave, and Scilab. The NumPy stack is also sometimes referred to as the SciPy stack.

Command - pip install scipy

(iv) OpenCV:

OpenCV is a library of programming functions mainly aimed at real time computer vision. originally developed by Intel, it is later supported by Willow Garage then Itseez. The library is cross-platform and free to use under the open-source BSD license.

Command - pip install opencv-python

(v) Pillow:

Python Imaging Library is a free Python programming language library that provides support to open, edit and save several different formats of image files. Windows, Mac OS X and Linux are available for this.

Command - pip install pillow

(vi) Matplotlib:

Matplotlib is a Python programming language plotting library and its NumPy numerical math extension. It provides an object-oriented API to use general-purpose GUI toolkits such as Tkinter, wxPython, Qt, or GTK+ to embed plots into applications.

Command - pip install matplotlib

(vii) H5py:

The software h5py includes a high-level and low-level interface for Python's HDF5 library. The low interface is expected to be complete wrapping of the HDF5 API, while the high-level component uses established Python and NumPy concepts to support access to HDF5 files, datasets and groups.

A strong emphasis on automatic conversion between Python (Numpy) data types and data structures and their HDF5 equivalents vastly simplifies the process of reading and writing data from Python.

Command - pip install h5py

(viii) Keras:

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

Command - pip install keras

(ix) ImageAI:

ImageAI provides an API to recognize 1000 different objects in a picture using pre-trained models that were trained on the ImageNet-1000 dataset. The model implementations provided are SqueezeNet, ResNet, InceptionV3 and DenseNet.

Command - pip3 install imageai --upgrade

3) Download the Yolo model file that will be used for object detection

3.5 Operational Feasibility Study

The vehicle has two video input streams as there are two sensors, one on each side of the vehicle. Both sensors are considered identical so both video streams are homogeneous. The solution must meet the constraints of a specific sonar configuration, but it will be shown that the throughput is large enough to handle any realistic sonar configuration. The sensors have a rather low throughput, near 13.5 new lines every second. Each line is 4166 pixels wide, which results in large images. Since the vehicle goes forward in its environment and periodically prepends pixel lines to both images, the images, here referred to as scrolling images, have a potentially infinite height. One of the axes of decision is thus the number of lines to be kept for processing, while maintaining a balance between detection rate and efficiency. Indeed, large objects would hardly be detected if the image was exceedingly small and computation might need more memory than the actual storage size if the image was too large. In addition, object detection is not to be performed for each single new line but may be performed when a sufficient number of lines have been prepended instead. It then enables the system to have a lower throughput and to run less frequently and it is expected to give the same results. From these requirements, a certain number of criteria can be drawn. First of all, the throughput requirement has an impact on how many lines are to be prepended between two object detection runs. Hence throughput is one of the axes solution evolution was based on. Since the platform adapts its power consumption in an autonomous manner against the resources that are required by user application, it is interesting to minimize resource requirements. Resource refers to both memory footprint and computation footprint (number of required cores and warps). In addition, as a future work new tasks might be added to be performed concurrently with object detection. Hence it makes sense to find a solution that matches all constraints while not using the full hardware potential. Hardware architecture can be taken advantage of, for instance by using CUDA streams to be scheduled onto both stream multiprocessors of the platform.

CHAPTER 4

SYSTEM DESIGN

4.1 Pictorial Diagram

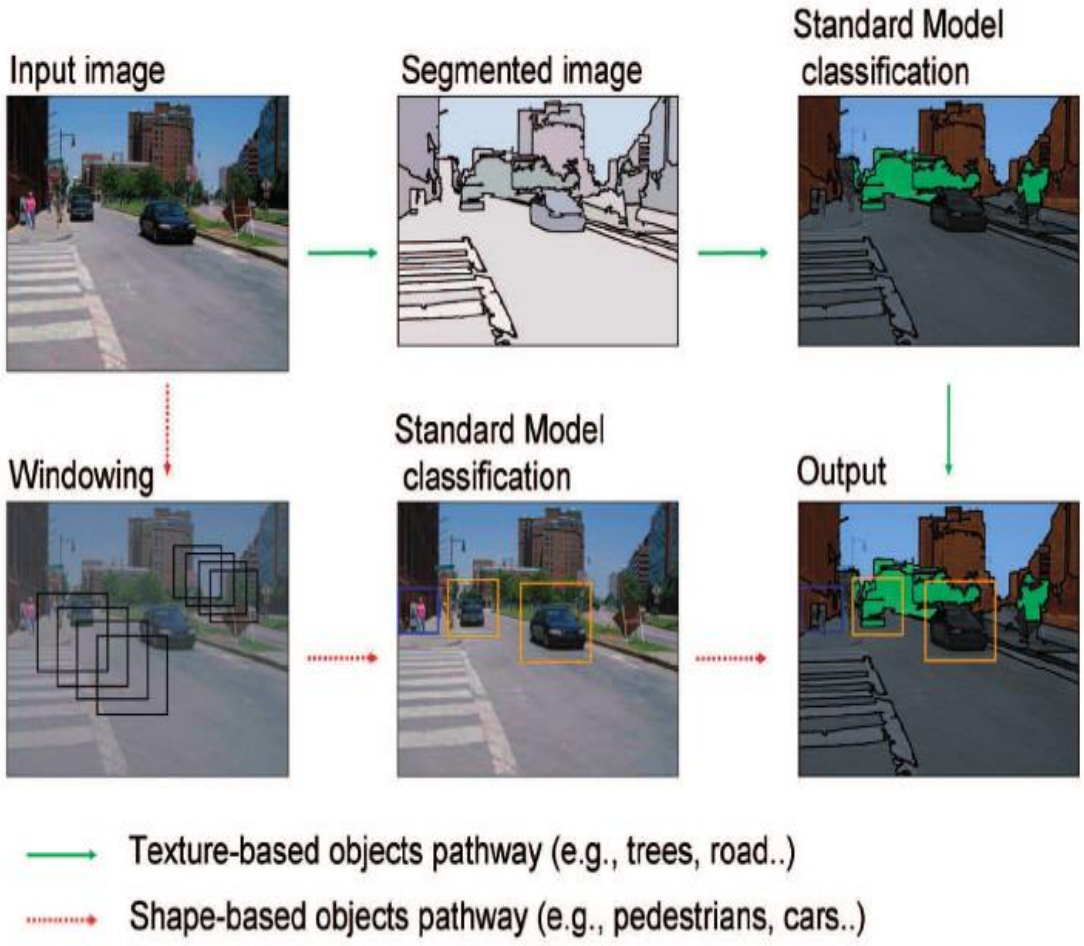


Figure 4.1: Pictorial diagram showing Object Detection on an image

The above DFD or Data Flow Diagram shows the flow of data through the entire process of object detection. It breaks down the process of classifying an object into the classes as specified in the model during the process of training.

4.2. 0- Level Data Flow Diagram

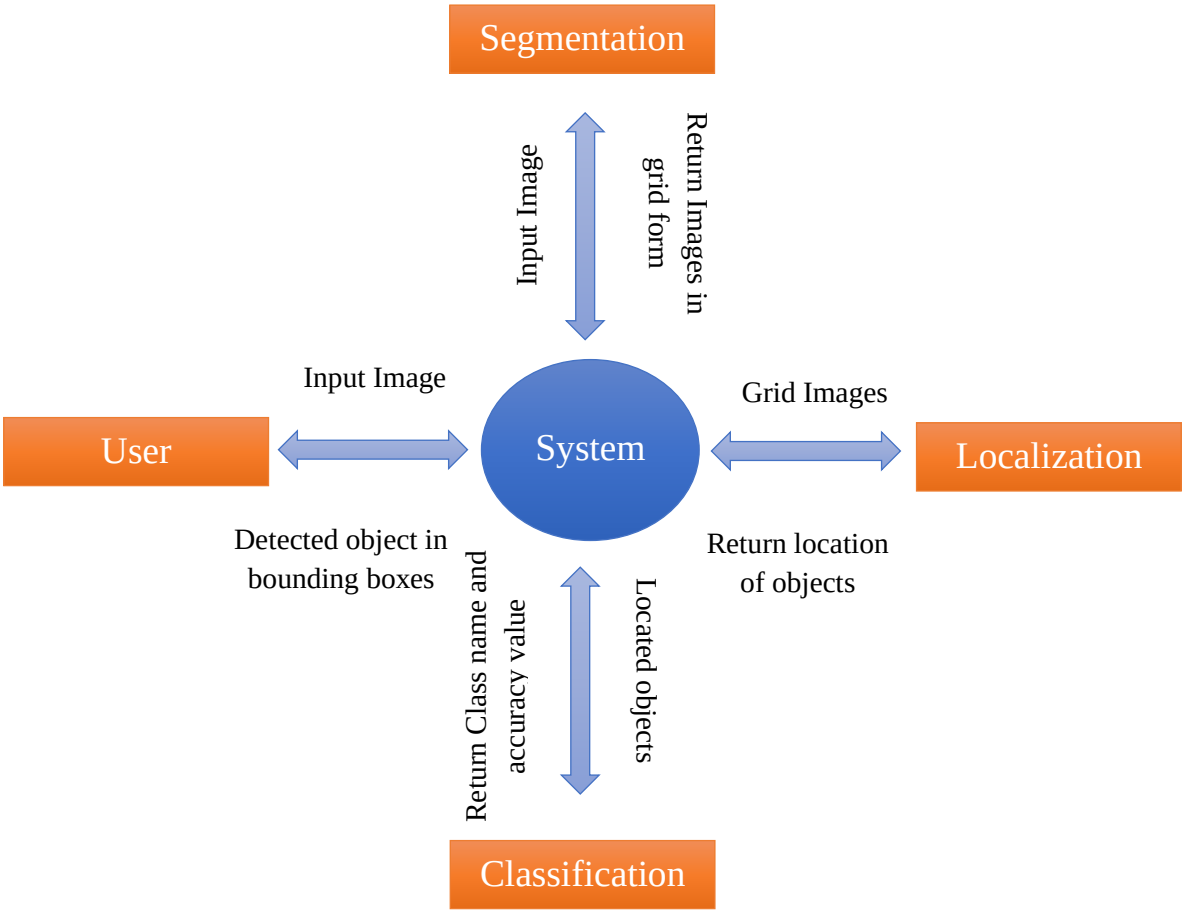


Figure 4.2: O - *Level DFD for Object Detection*

1 - Level Data Flow Diagram

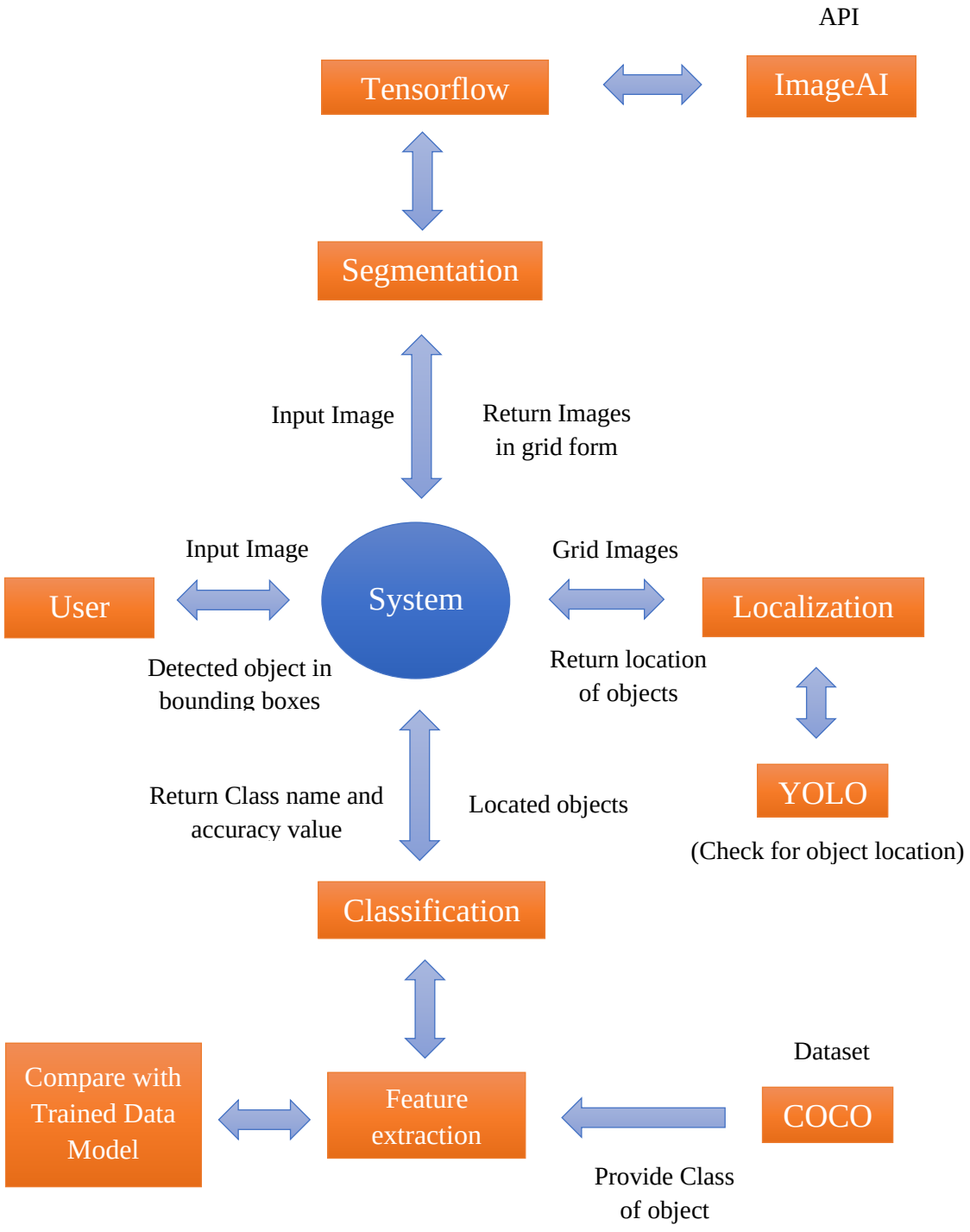


Figure 4.3: 1- *Level DFD for Object Detection*

CHAPTER 5

TESTING

5.1 About Testing

Various set of images, video and camera (as webcam) were used and fed as input data to our traning model. The output was then obtained consisting of bounding boxes around the detected object. These samples were then used as test cases for verifying and testing the accuracy and reliability of our model in detecting object.

5.2 Test Cases

Test Case 1

Project Name: OBJECT DETECTION AND IDENTIFICATION

Test Case ID: cam

Test Priority (Low/Medium/High): Med

Test Title: Identify the objects in the input image

Test Executed by: Ayush Maheshwari

Test Execution date: 25/04/2020

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to code	Image to be tested	Bounding boxes containing name of the object in the image	Bounding box containing objects name book, person has been identified	Pass	
2	Provide path of image to be tested					
3	Provide the name of output image					
4	Run the code					

Test Case 2

Project Name: OBJECT DETECTION AND IDENTIFICATION

Test Case ID: video

Test Priority (Low/Medium/High): Med

Test Title: Identify the objects in the input image

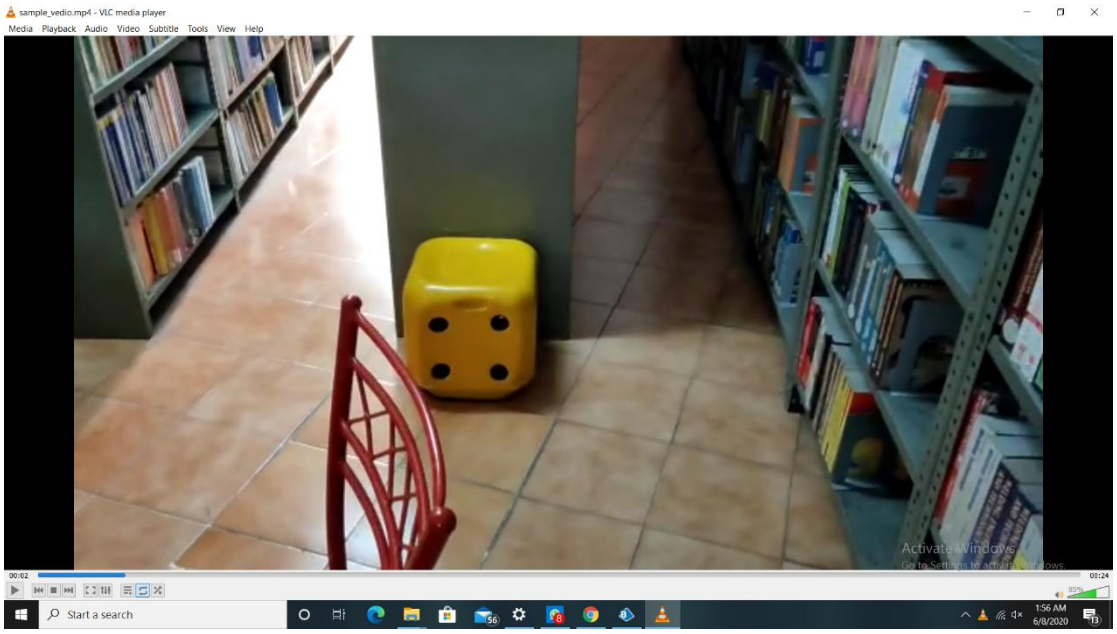
Test Executed by: Kundan Jha

Test Execution date: 25/04/2020

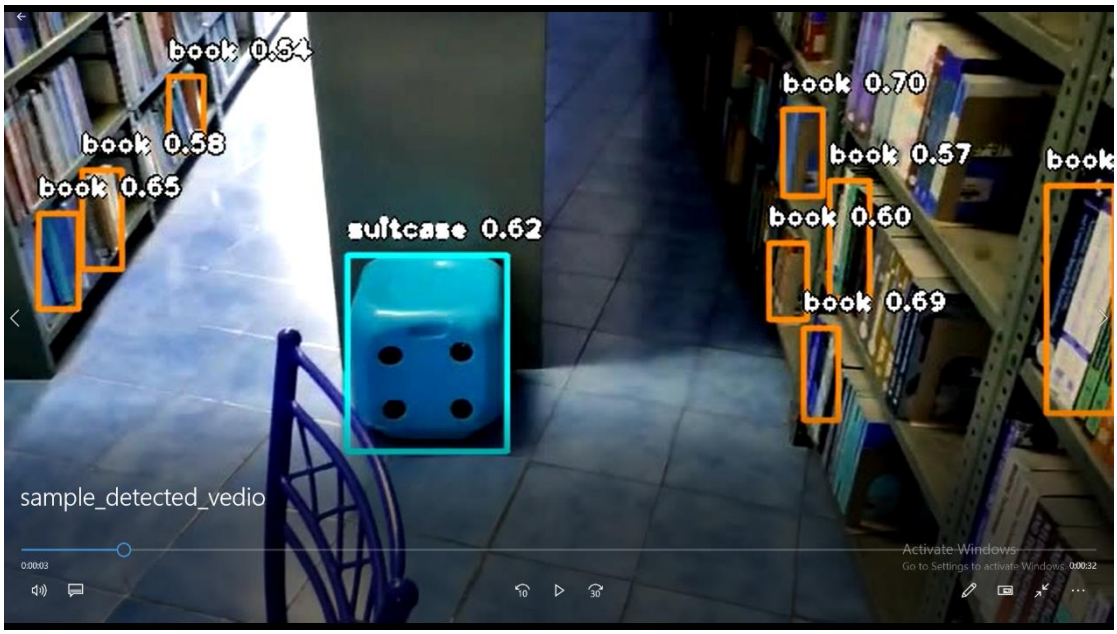
Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to code	Image to be tested	Bounding boxes containing name of the object in the image	Bounding box containing objects name book, suitcase has been identified	Fail	
2	Provide path of image to be tested					
3	Provide the name of output image					
4	Run the code					

Test Id: video

Input Image



Output Image:



Test Case 3

Project Name: OBJECT DETECTION AND IDENTIFICATION

Test Case ID: img2

Test Priority (Low/Medium/High): Med

Test Title: Identify the objects in the input image

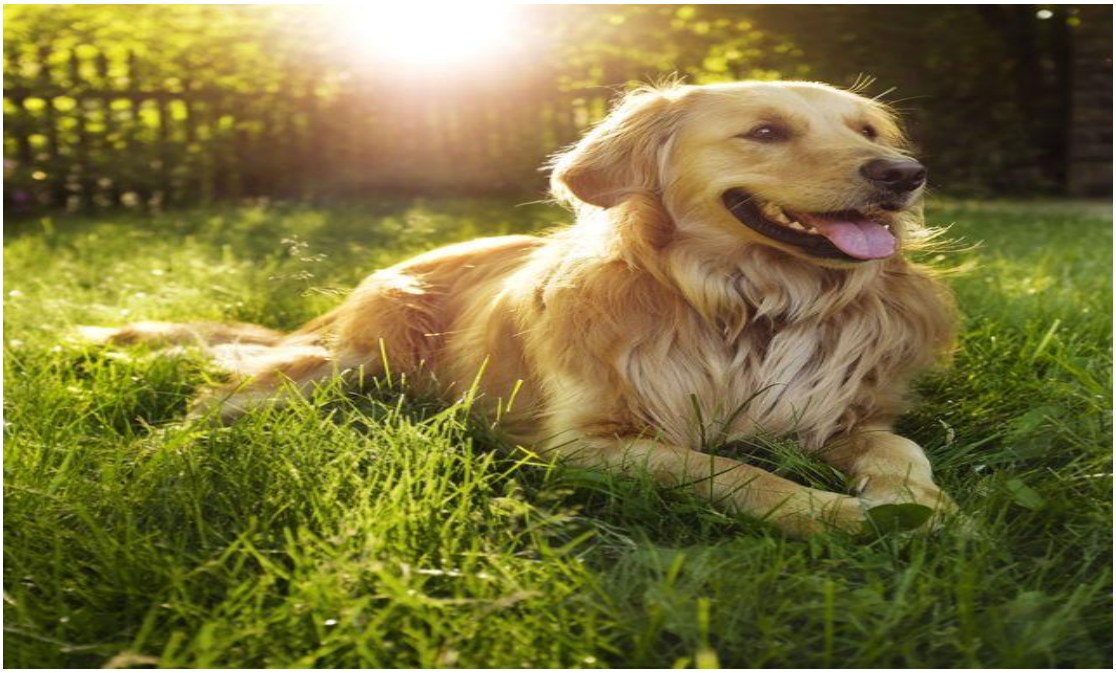
Test Executed by: Ayush Maheshwari

Test Execution date: 02/04/2020

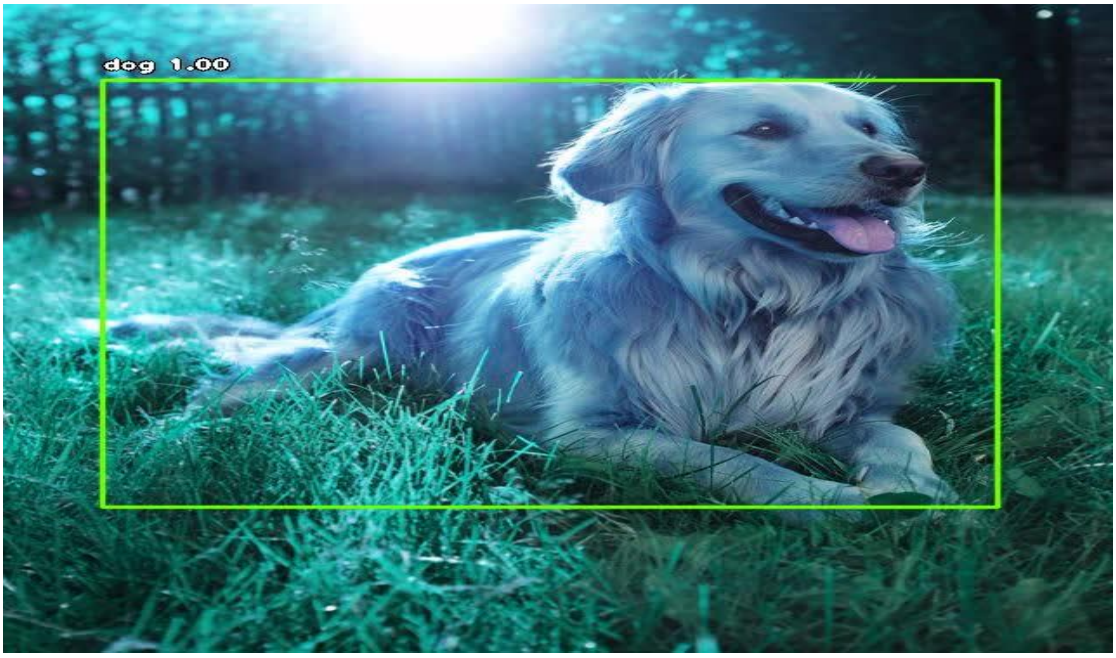
Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to code	Image to be tested	Bounding boxes containing name of the object in the image	Bounding box containing object name dog has been identified	Pass	
2	Provide path of image to be tested					
3	Provide the name of output image					
4	Run the code					

Test Id: img2

Input Image:



Output Image:



Test Case 4

Project Name: OBJECT DETECTION AND IDENTIFICATION

Test Case ID: img3

Test Priority (Low/Medium/High): Med

Test Title: Identify the objects in the input image

Test Executed by: Harsh Nain

Test Execution date: 06/04/2020

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to code	Image to be tested	Bounding boxes containing name of the object in the image	Bounding box containing objects name dog, person, car has been identified	Pass	
2	Provide path of image to be tested					
3	Provide the name of output image					
4	Run the code					

Test Id: img 3

Input Image

Output Image

Test Case 5

Project Name: OBJECT DETECTION AND IDENTIFICATION

Test Case ID: img4

Test Priority (Low/Medium/High): Med

Test Title: Identify the objects in the input image

Test Executed by: Kundan Jha

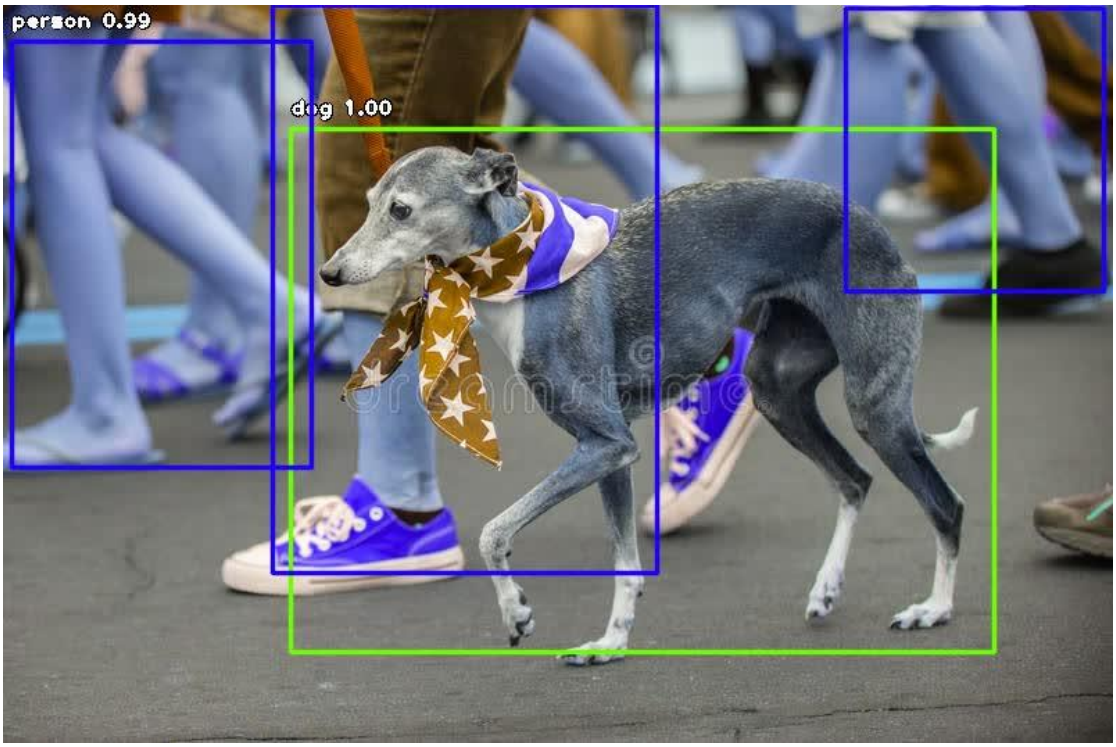
Test Execution date: 13/04/2020

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to code	Image to be tested	Bounding boxes containing name of the object in the image	Bounding box containing objects name dog, person has been identified	Pass	
2	Provide path of image to be tested					
3	Provide the name of output image					
4	Run the code					

Test Id: img4

Input Image:

Output Image:



Test Case 6

Project Name: OBJECT DETECTION AND IDENTIFICATION

Test Case ID: img5

Test Priority (Low/Medium/High): Med

Test Title: Identify the objects in the input image

Test Executed by: Divyanshi Agarwal

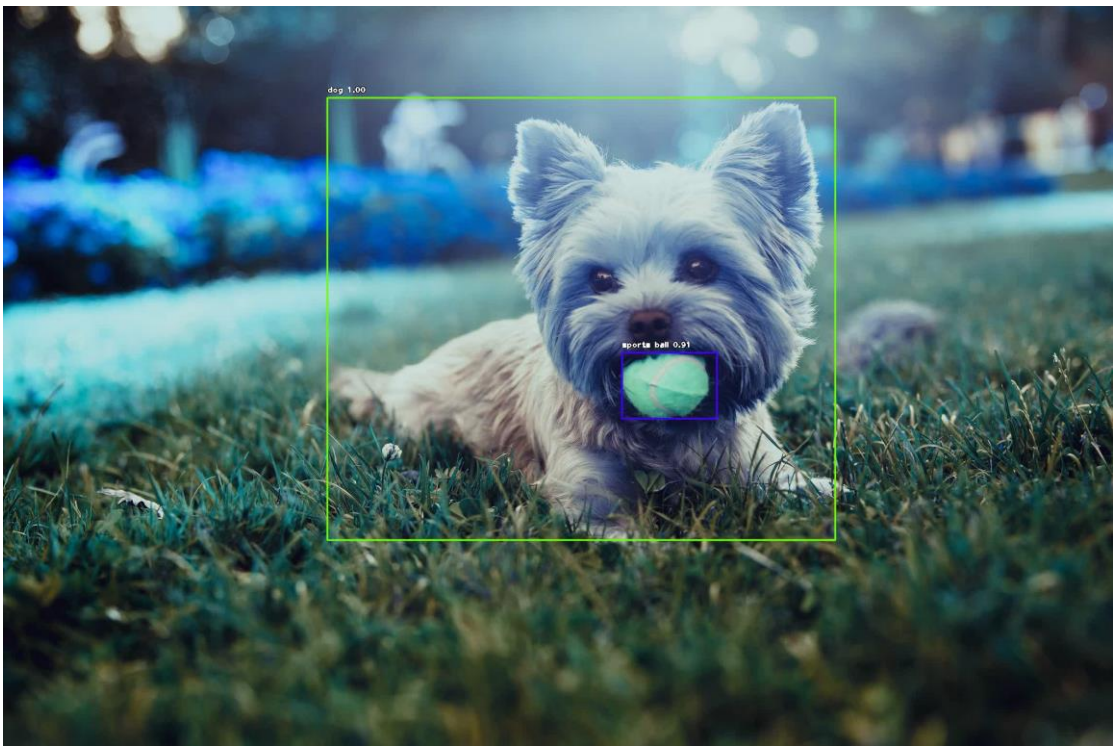
Test Execution date: 20/04/2020

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to code	Image to be tested	Bounding boxes containing name of the object in the image	Bounding box containing objects name dog, sports ball has been identified	pass	
2	Provide path of image to be tested					
3	Provide the name of output image					
4	Run the code					

Test Id: img5

Input Image:

Output Image:



CHAPTER 6

SNAPSHOTS

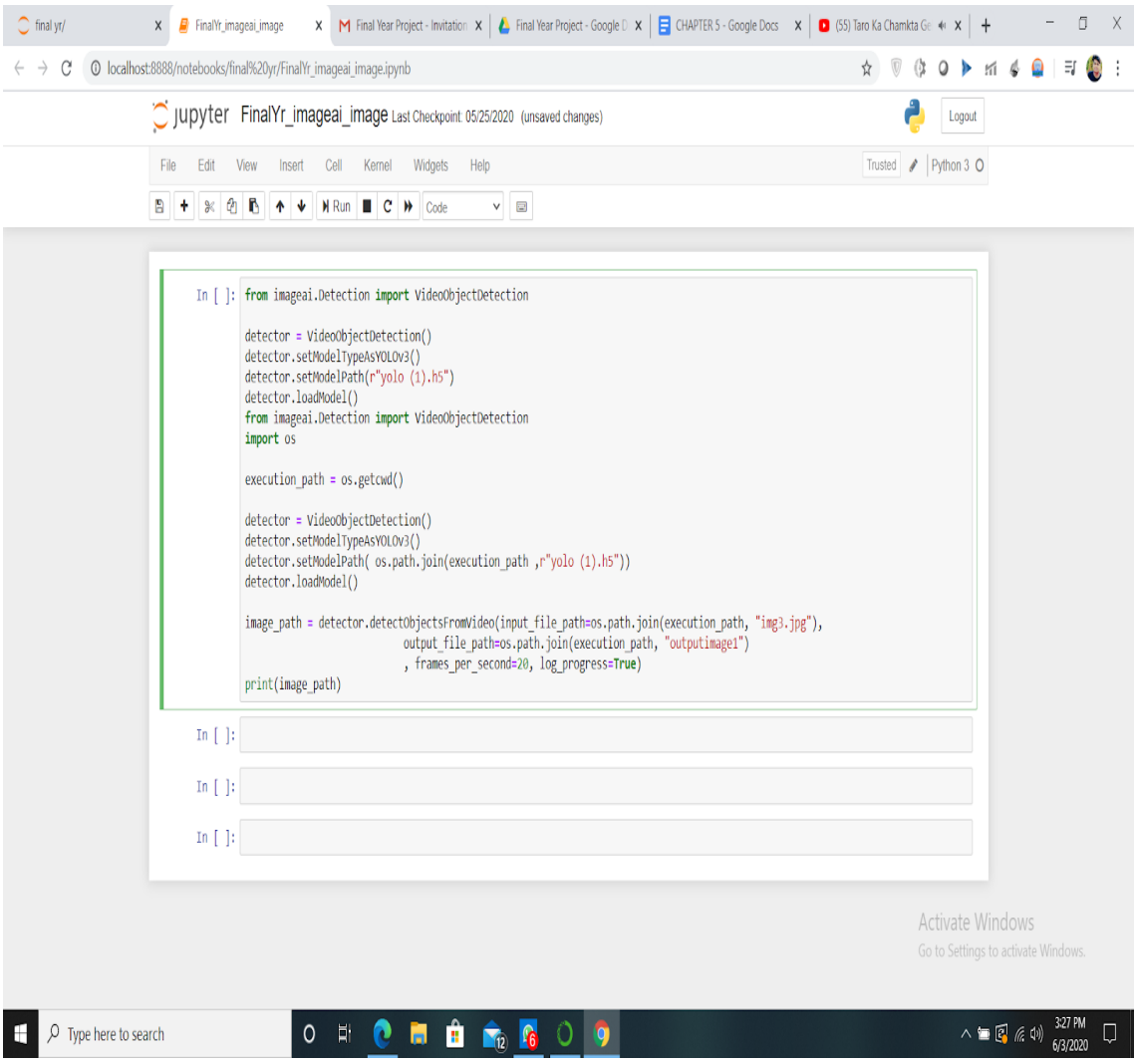


Figure 6.1: Jupyter Notebook showing the code for object detection using ImageAI (for image)

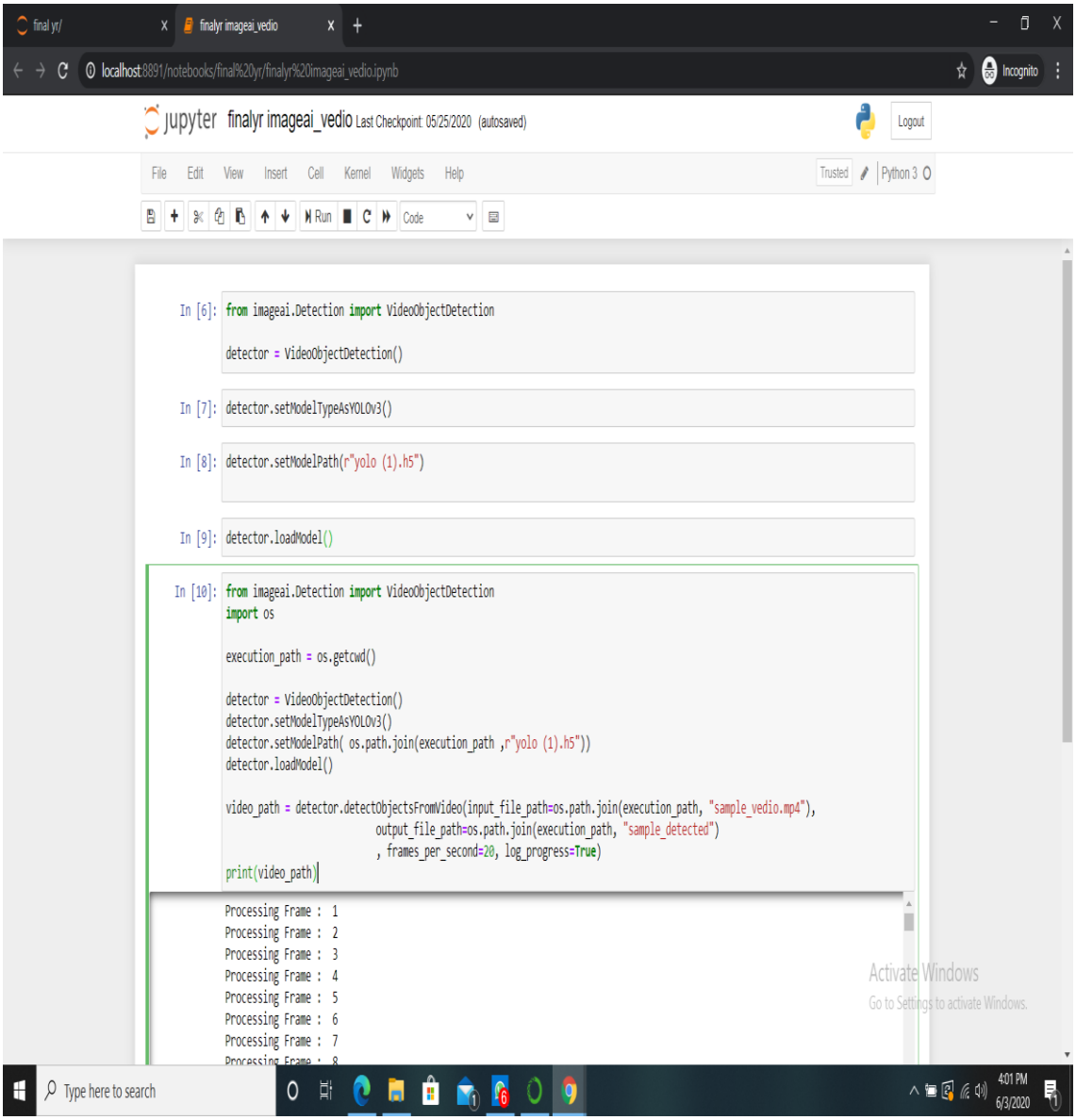


Figure 6.2: Jupyter Notebook showing the code for object detection using ImageAI (for video)

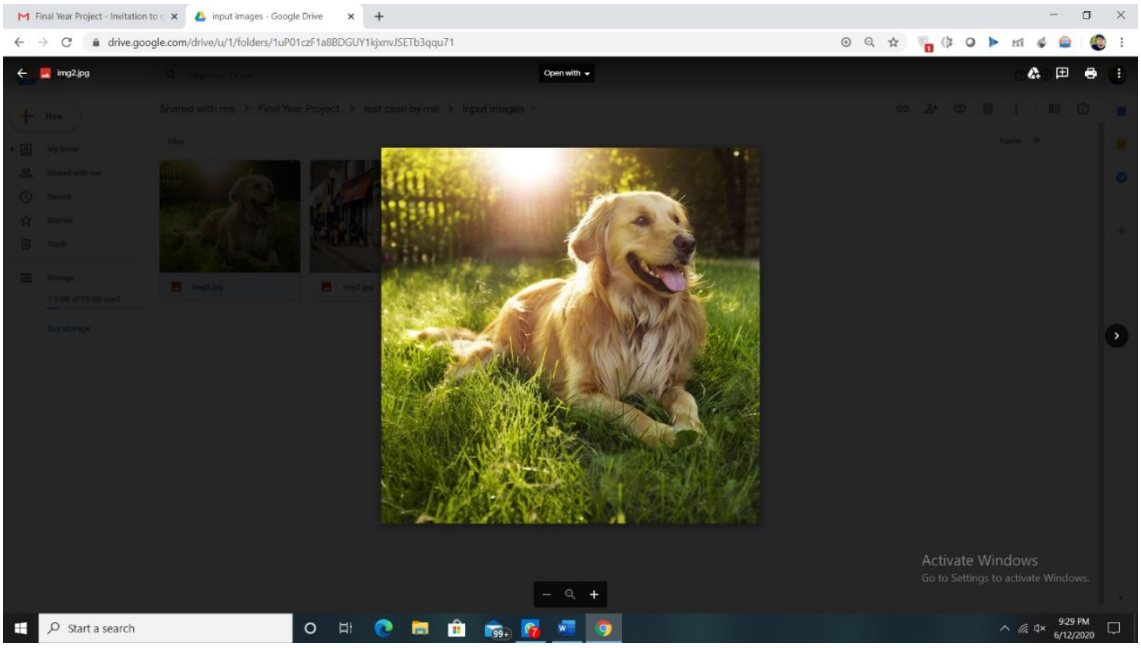


Figure 6.3: Input image containing dog

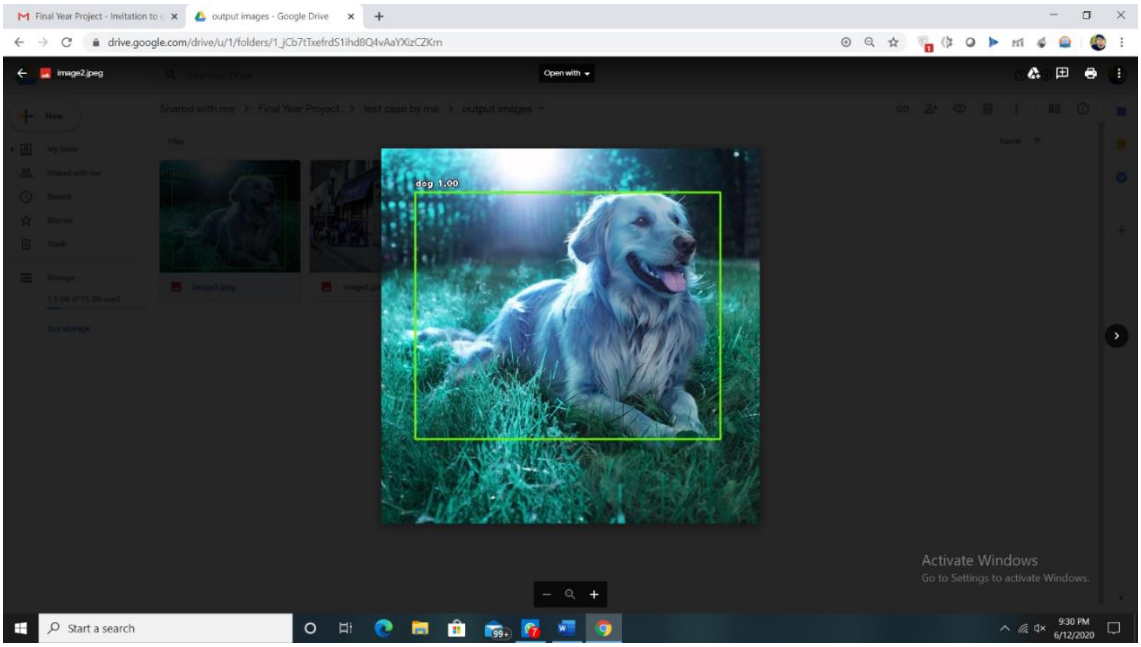


Figure 6.4: Output image show bounding box for dog

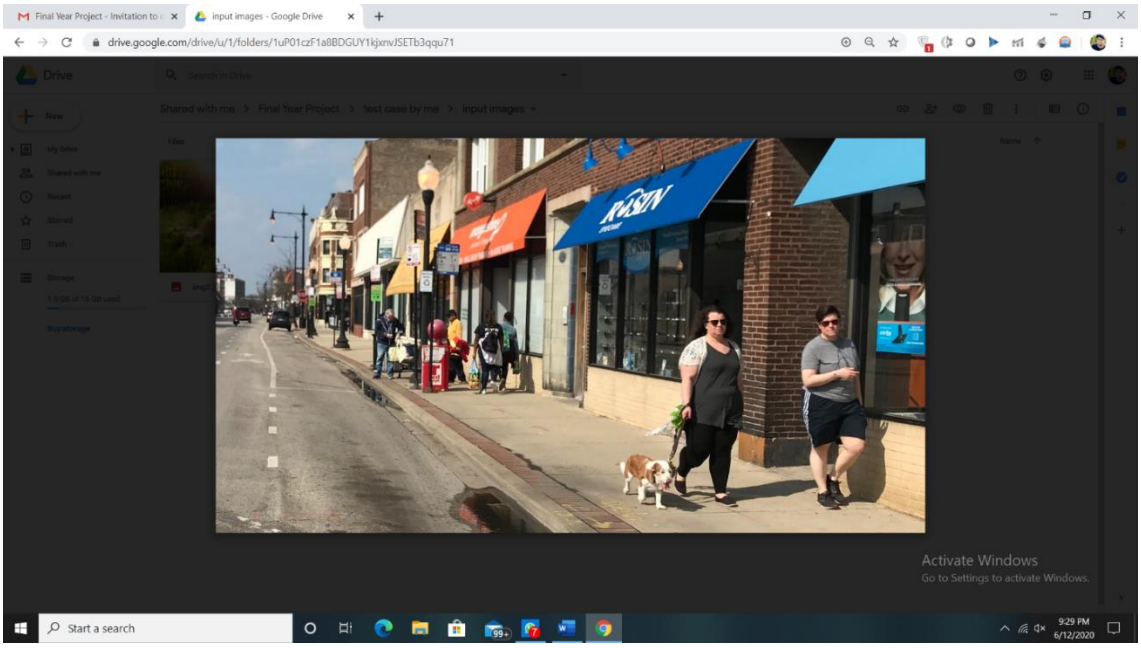


Figure 6.5: Input image containing dog, person, cars, etc.

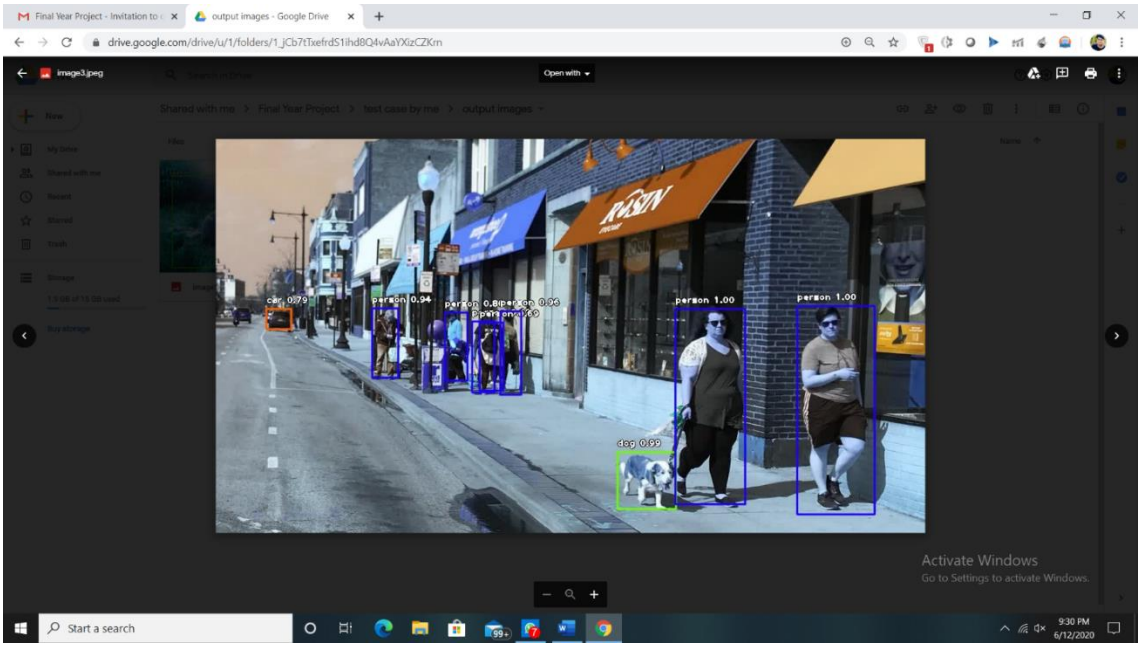


Figure 6.6: Output image show bounding box for dog, person, cars, etc.

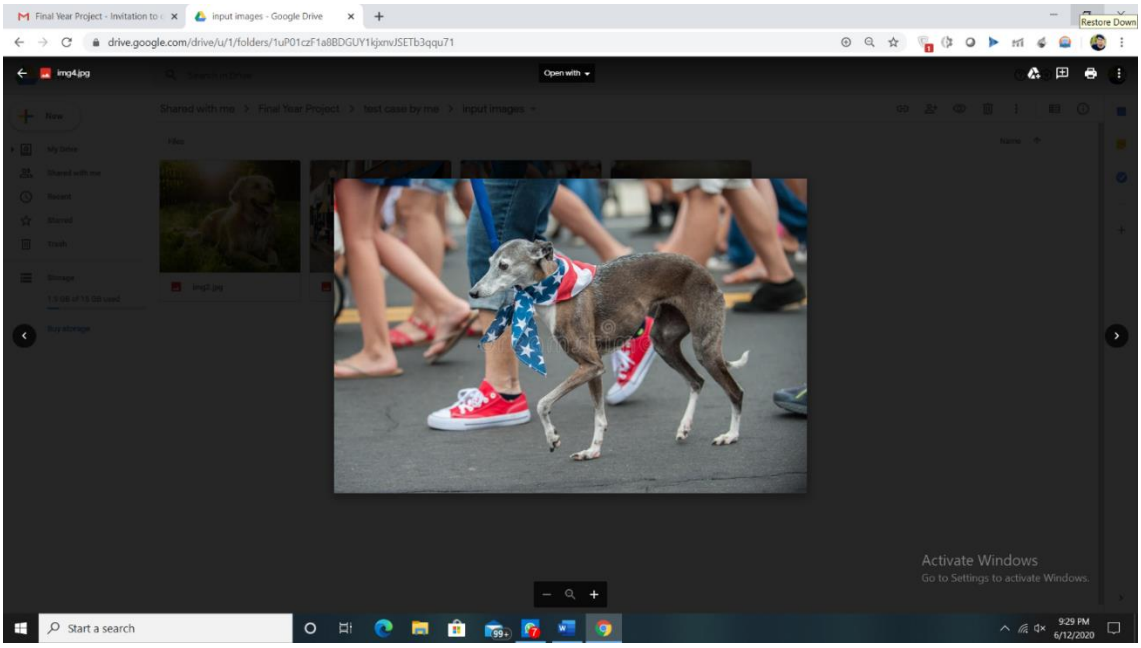


Figure 6.7: Input image show bounding boxes for dog, person, etc.

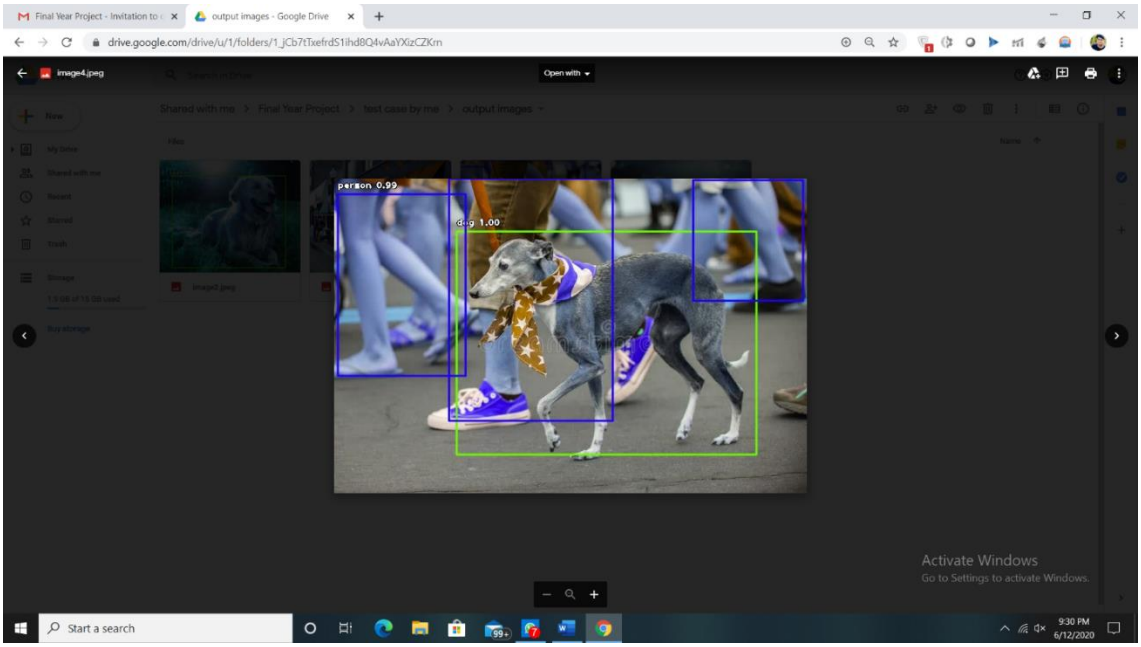


Figure 6.8: Output image show bounding boxes for dog, person, etc.

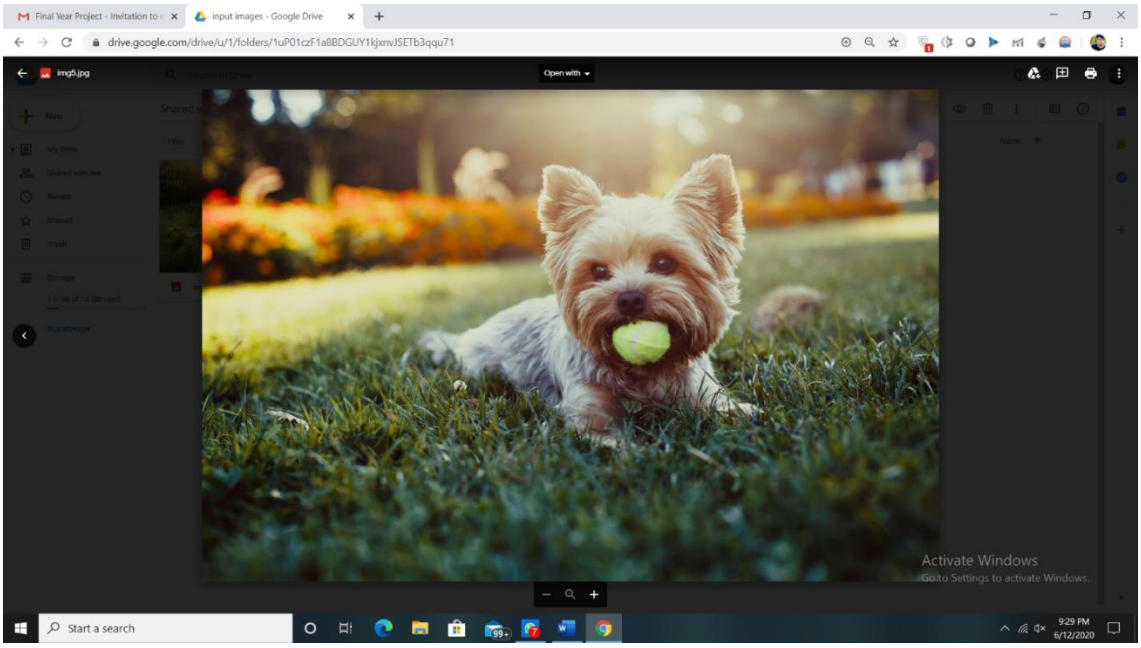


Figure 6.9: Input image containing dog and sports ball

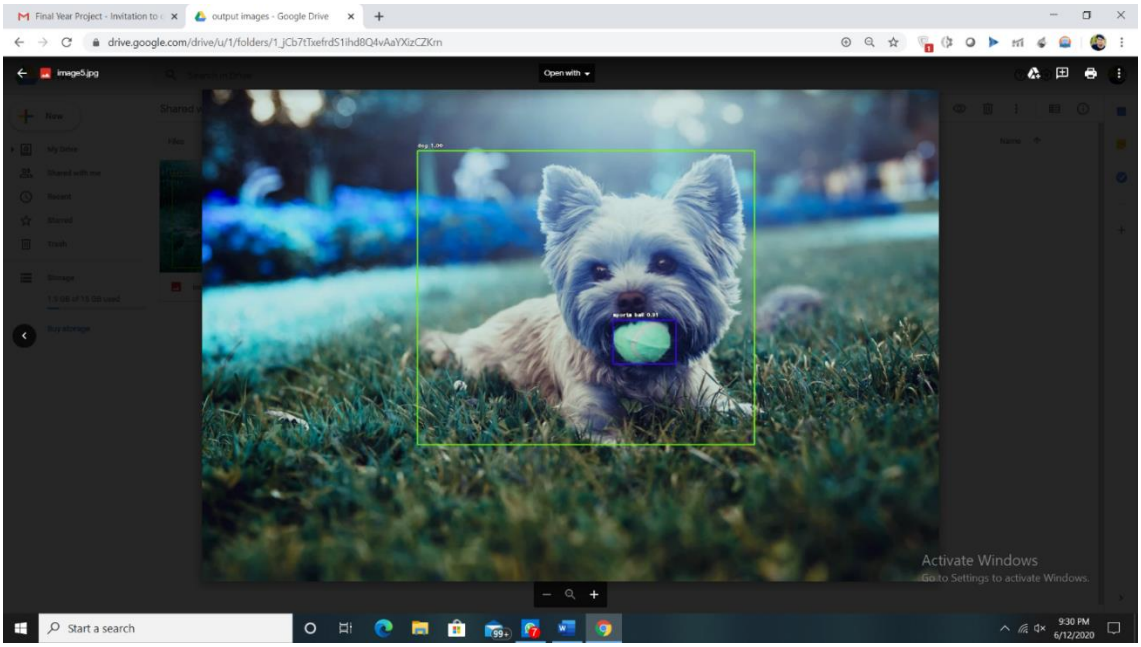


Figure 6.10: Output image show bounding box for dog and ball

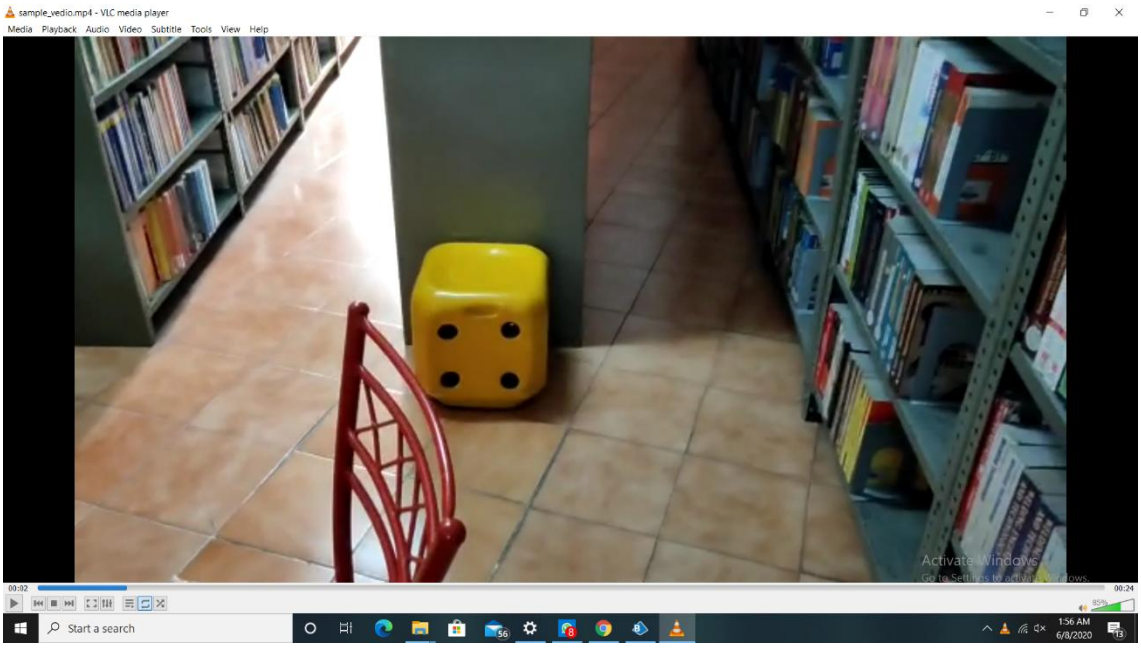


Figure 6.11: Input video for testing

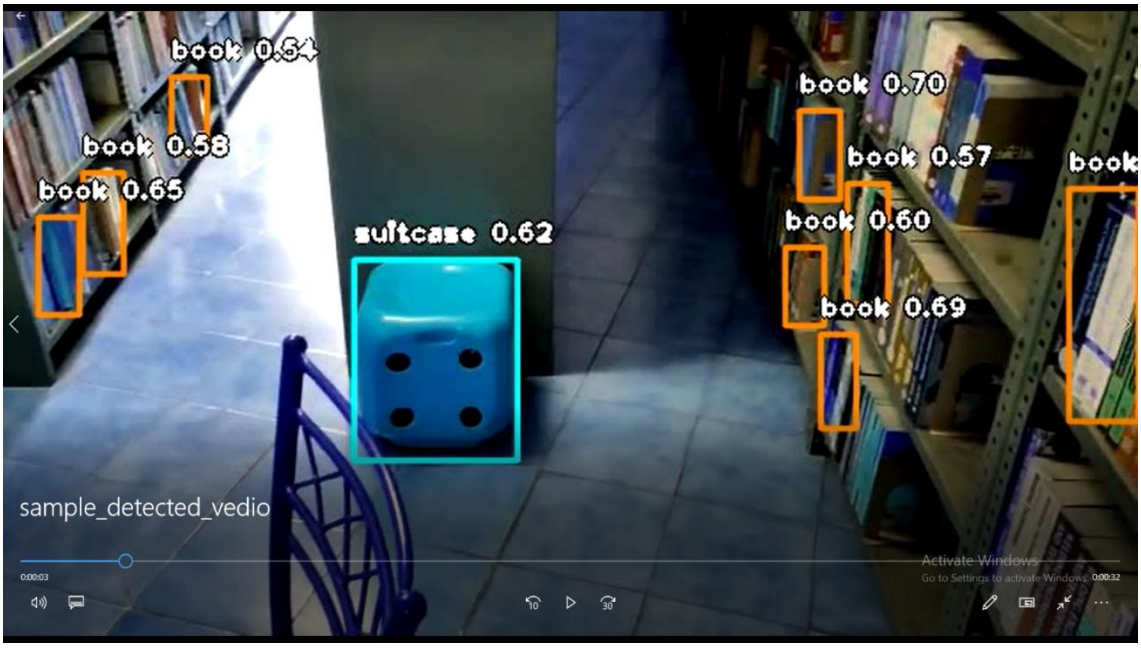


Figure 6.12: Output video with detected object after testing

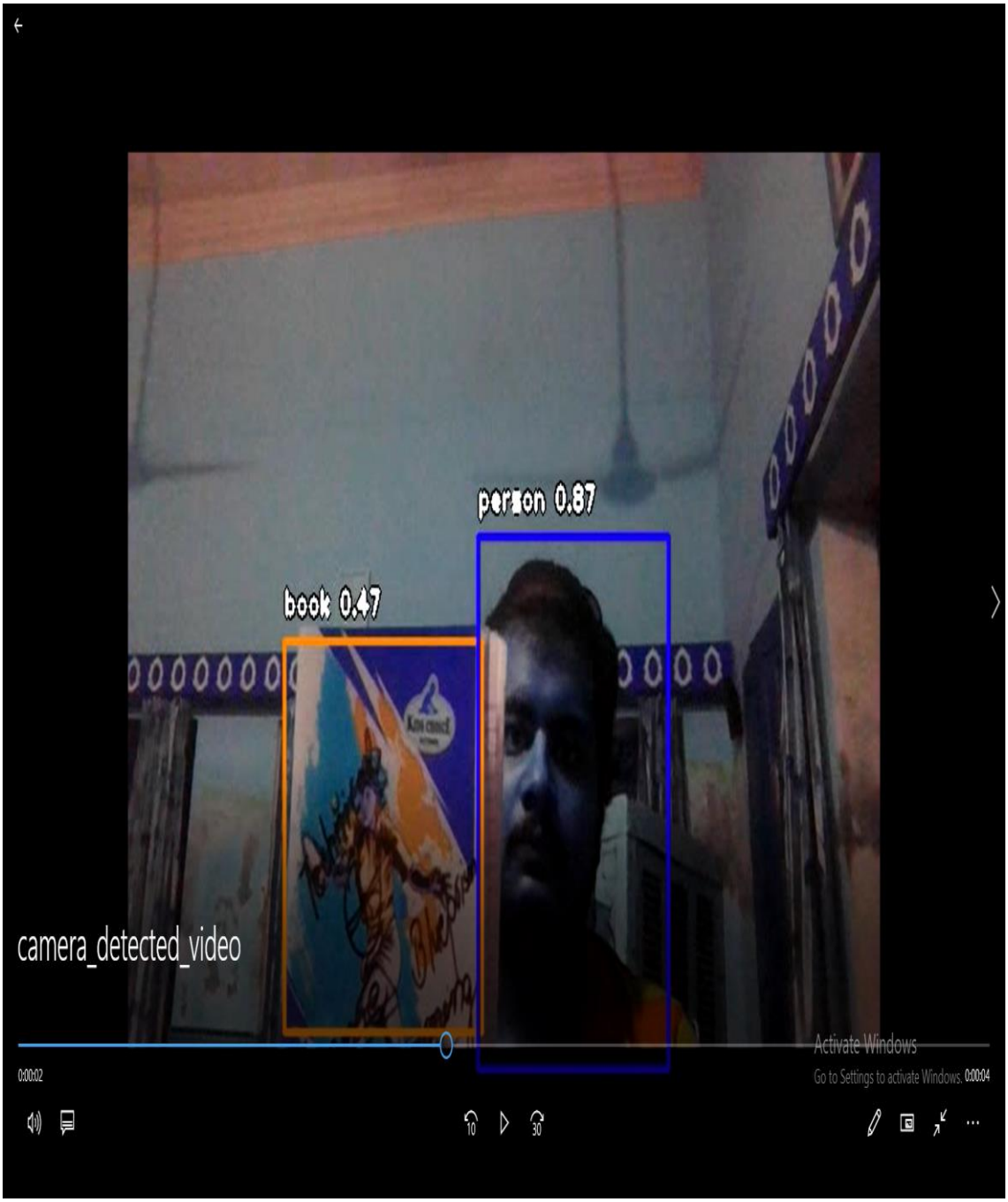


Figure 6.13: Input from webcam and output after testing

CHAPTER 7

CONCLUSION

By using this thesis and based on experimental results we are able to detect objects more precisely and identify the objects individually with the exact location of an object in the picture in the x, y axis. We have also accomplished **Video object detection** using ImageAI by using its pre-trained models.

7.1 Limitations of the project-

Viewpoint Variation: In a real world, the entities within the image are aligned in different directions and when such images are fed to the system, the system predicts inaccurate values. In short, the system fails to understand that changing the alignment of the image (left, right, bottom, top) will not make it different and that is why it creates challenges in image recognition.

1) **Deformation:**

   Objects do not change even if they are deformed. The system learns from the perfect image and forms a perception that a particular object can be in specific shape only. We know that in the real world, shape changes and as a result, there are inaccuracies when the system encounters a deformed image of an object.

2) **Inter-class Variation:**

   Certain objects vary within the class. They can be of different shape, size, but still represent the same class. For example, buttons, chairs, bottles, bags come in different sizes and appearances.

3) **Multi-class**:

Many applications require detecting more than one object class. If a large number of classes are being detected, the processing speed becomes an important issue, as well as the kind of classes that the system can handle without accuracy loss.

4) **Contextual Information and Temporal Features:**

Integrating contextual information (e.g., about the type of scene, or the presence of other objects) can increase speed and robustness, but "when and how" to do this (before, during or after the detection); it is still an open problem.

5) **Occlusions, Deformable Objects, and Interlaced Objects and Background:**

Dealing with partial occlusions is also an important problem, and no compelling solution exists. Similarly, detecting objects that are not "closed," i.e., where objects and background pixels are interlaced with background is still a difficult problem.

7.2 Future Scope of the project

The object recognition system can be applied in the area of surveillance system, face recognition, fault detection, character recognition etc. The objective of this project is to develop an object detection system to recognize the 2D and 3D objects in the image as well as in videos.

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,

I. An object recognition system is developed that recognizes the two-dimensional and three-dimensional objects.
II. 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.
III. The proposed global feature extraction requires less time, compared to the traditional feature extraction method.

As a scope for future enhancement

I. Features either the local or global used for recognition can be increased, to increase the efficiency of the object recognition system.
II. Geometric properties of the image can be included in the feature vector for recognition.
III. Using an unsupervised classifier instead of a supervised classifier for recognition of the object.

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:

I. 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.
II. Fully occluded objects cannot be tracked and considered as a new object in the next frame.

For Night time visual tracking, night vision mode should be available as an inbuilt feature in the CCTV camera.

I. An important problem is to incrementally learn, to detect new classes, or to incrementally learn to distinguish among subclasses after the “main” class has been learned. If this can be done in an unsupervised way, we will be able to build new classifiers based on existing ones, without much additional effort, greatly reducing the effort required to learn new object classes. Note that humans are continuously inventing new objects, fashion changes, etc., and therefore detection systems will need to be continuously updated, adding new classes, or updating existing ones.

References

Books

1. Agarwal, S., Awan, A., and Roth, D. (2004). Learning to detect objects in images via a sparse, part-based representation. IEEE Trans. Pattern Anal. Mach. Intell. 26,1475–1490.

   doi:10.1109/TPAMI.2004.108

2. Alexe, B., Deselaers, T., and Ferrari, V. (2010). "What is an object?" in Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on (San Francisco, CA: IEEE), 73–80.

   doi:10.1109/CVPR.2010.5540226

3. Aloimonos, J., Weiss, I., and Bandyopadhyay, A. (1988). Active vision. Int. J. Comput. Vis. 1,

   333–356. doi:10.1007/BF00133571

4. Andreopoulos, A., and Tsotsos, J. K. (2013). 50 years of object recognition: directions forward. Comput. Vis. Image Underst. 117, 827–891. doi: 10.1016/j.cviu.2013.04.005

5. Azizpour, H., and Laptev, I. (2012). "Object detection using strongly-supervised deformable part models," in Computer Vision-ECCV 2012 (Florence: Springer),836–849.

6. Azzopardi, G., and Petkov, N. (2013). Trainable cosfire filters for keypoint detection and pattern recognition. IEEE Trans. Pattern Anal. Mach. Intell. 35, 490–503.doi:10.1109/TPAMI.2012.106

7. Azzopardi, G., and Petkov, N. (2014). Ventral-stream-like shape representation:from pixel intensity values to trainable object-selective cosfire models. Front.Comput. Neurosci. 8:80.

   doi:10.3389/fncom.2014.00080

8. Benbouzid, D., Busa-Fekete, R., and Kegl, B. (2012). "Fast classification using sparse decision dags," in Proceedings of the 29th International Conference on MachineLearning (ICML-12), ICML '12, eds

   J. Langford and J. Pineau (New York, NY:Omnipress), 951–958.

9. Bengio, Y. (2012). "Deep learning of representations for unsupervised and transfer learning," in ICML Unsupervised and Transfer Learning, Volume 27 of

JMLRProceedings, eds I. Guyon, G. Dror, V. Lemaire, G. W. Taylor, and D. L. Silver (Bellevue: JMLR.Org), 17–36.

10. Bourdev, L. D., Maji, S., Brox, T., and Malik, J. (2010). “Detecting people using mutually consistent poselet activations,” in Computer Vision – ECCV2010 – 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part VI, Volume 6316 of Lecture Notes in Computer Science, eds K. Daniilidis, P. Maragos, and N. Paragios

    (Heraklion:Springer), 168–181.

11. Bourdev, L. D., and Malik, J. (2009). “Poselets: body part detectors trained using 3dhuman pose annotations,” in IEEE 12th International Conference on ComputerVision, ICCV 2009, Kyoto, Japan,

    September 27 – October 4, 2009 (Kyoto: IEEE),1365–1372.

12. Cadena, C., Dick, A., and Reid, I. (2015). “A fast, modular scene understanding system using context-aware object detection,” in Robotics and Automation (ICRA),2015 IEEE International Conference on (Seattle, WA).

13. Correa, M., Hermosilla, G., Verschae, R., and Ruiz-del-Solar, J. (2012). Human Detection and identification by robots using thermal and visual information in domestic environments. J. Intell.

    Robot Syst. 66, 223–243. doi:10.1007/s10846-011-9612-2

14. Dalal, N., and Triggs, B. (2005). “Histograms of oriented gradients for human detection,” in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEEComputer Society Conference on,

    Vol. 1 (San Diego, CA: IEEE), 886–893. doi:10.1109/CVPR.2005.177

15. Erhan, D., Szegedy, C., Toshev, A., and Anguelov, D. (2014). “Scalable object detec-tion using deep neural networks,” in Computer Vision and Pattern Recognition Frontiers in Robotics and AI

    www.frontiersin.org November 2015

Websites:

- Object Detection:
  https://en.wikipedia.org/wiki/Object_detection

- Image AI GitHub Repository:
  https://github.com/OlafenwaMoses/ImageAI#videodetection

- Object Detection From The Satellite Images Using Kaur:
  https://www.semanticscholar.org/paper/Object-Detection-from-the-Satellite-Images-Using-Kaur/602ff4fd0f5bd10c9fb971ecd2317e542f070883

- Video and Live-Feed Detection and Analysis:
  https://imageai.readthedocs.io/en/latest/video/

- Object detection speed and accuracy comparison:
  https://mc.ai/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo/

- Understanding image recognition and its uses:
  https://www.einfochips.com/blog/understanding-image-recognition-and-its-uses/

- Object Detection: Current and Future Directions
  https://www.frontiersin.org/articles/10.3389/frobt.2015.00029/full

- Object Detection:
  https://www.cse.iitb.ac.in/~pratikm/projectPages/objectDetection/downloads/report.pdf