VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Indoor Object Detection and Identification through Webcam

Submitted towards Project Review for the course

IMAGE PROCESSING

CSE4019

Under the guidance of

Prof. Swathi JN

Submitted by

18BCE2511- RITIKA MANDAL
18BCI0148- JAI KATHURIA
18BCI0131 - RAGHAV SOMANI

Acknowledgement

We take this opportunity to express our profound gratitude and deep regards to our Professor Prof.Swathi J.N for her exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by her time to time shall carry us a long way in the journey of life .

Executive Summary

Object detection is so important in the world right now as it is used in many fields like Healthcare, Agriculture, Autonomous Driving, and more. It provides an efficient way of handling image classification by detecting the object in the image and letting us know where it is in the image using localization,i.e, it creates a bounding box around the object. 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. A computer views all kinds of visual media as an array of numerical values. As a consequence of this approach, they require image processing algorithms to inspect the contents of images. This project compares two major image processing algorithms: Single Shot Detection (SSD), and You Only Look Once (YOLO) to find the fastest and most efficient of both. In this comparative analysis, using the Microsoft COCO (Common Object in Context) dataset, the performance of these two algorithms is evaluated and their strengths and limitations are analysed based on parameters such as accuracy, precision and F1 score. From the results of the analysis, it can be concluded that the suitability of any of the algorithms over the other is dictated to a great extent by the use cases they are applied in. In an identical testing environment, YOLO-v3 outperforms SSD, making it the best of the two algorithms. Single Shot MultiBox Detector is a deep learning model used to detect objects in an image or from a video source. Single Shot Detector is a simple approach to solve the problem but it is very effective till now. YOLO uses a totally different approach. YOLO is a clever convolutional neural network (CNN) for doing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

Table of Contents

List of Figures

List of Tables

Abbreviations

SSD (Single Shot Detector) , YOLO (You Only Look Once) ,COCO (Common Objects in Context)

1. INTRODUCTION

1. Objective

Humans can detect and identify objects present in an image. The human visual system is fast and accurate and can also perform complex tasks like identifying multiple objects and detect obstacles with little conscious thought. The availability of large sets of data, faster GPUs, and better algorithms, we can now easily train computers to detect and classify multiple objects within an image with high accuracy.
The aim of indoor object detection and identification is to detect all instances of objects from a known class, such as people, toys 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 identification 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 for 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.

We need to understand terms such as object detection, object localization for object detection and recognition, and finally explore an object detection algorithm known as “You only look once” (YOLO).

2. Motivation

For a few decades, computer scientists and engineers have attached cameras and simplistic image interpretation methods to a computer in order to impart vision to the machine. A lot of interest has been shown towards object recognition, object detection, object categorization etc. Simply speaking, object recognition deals with training the computer to identify a particular object from various perspectives, in various lighting conditions, and with various backgrounds, object detection deals with identifying the presence of various individual objects in an image.
Great success has been achieved in an open environment for object detection/recognition problems but the problem remains unsolved in controlled and closed places, in particular, when objects are placed in arbitrary poses in cluttered and occluded environments. As an example, it might be easy to make algo to recognize the presence of a coffee machine with nothing else in the image. On the other hand imagine the difficulty of such algo in detecting the machine on a kitchen slab that is cluttered by other utensils, gadgets, tools, etc. The searching or recognition process in such scenarios is very difficult. So far, no effective solution has been found for this problem. Despite a lot of research in this area, the methods

developed so far are not efficient, require long training time, are not suitable for real time application, and are not scalable to large numbers of classes.
Any development of such capability “to detect and recognize objects in the indoor environment through the webcams” would bring us closer to realize our dream of employing a domestic help algorithm. The aim of this research is to develop a novel vision system that can do category object detection and recognition in a closed environment.

3. Background

Object Detection has found its application in a wide variety of domains such as video surveillance, image retrieval systems, autonomous driving vehicles and many more. Computer vision has a lot of interesting applications and object detection is one of the most interesting applications. With the advanced computer vision techniques, the objects present in the images can be identified in seconds with great accuracy. Hundreds of images can be processed in a few minutes to detect objects in those images. There are many algorithms available now through which this object detection can be performed very fastly. YOLO is one of these popular object detection methods.

YOLO vs. SSD

YOLO (You Only Look Once) system, an open-source method of object detection that can recognize objects in images and videos swiftly whereas SSD (Single Shot Detector) runs a convolutional network on input image only one time and computes a feature map. SSD is a better option as we are able to run it on a video and the exactness trade-off is very modest. SSD is a healthier recommendation. However, if exactness is not too much of disquiet but you want to go super quick, YOLO will be the best way to move forward. First of all, a visual thoughtfulness of swiftness vs precision trade-off would differentiate them well. While dealing with large sizes, SSD seems to perform well, but when we look at the accurateness numbers when the object size is small, the performance dips a bit.

2. PROJECT DESCRIPTION AND GOALS

Image classification and detection are the most important pillars of object detection. There is a plethora of datasets available. COCO is one such widely used image classification domain. It is a benchmark dataset for object detection. It introduces a large-scale dataset that is available for image detection and classification . This project aims to make a comparative analysis of SSD and YOLO. The first algorithm for the comparison in the current work is SSD which adds layers of several features to the end network and

facilitates ease of detection. While YOLO was developed by Joseph Redmon that offers an end-to-end network .

In this project, by using the COCO dataset as a common factor of the analysis and measuring the same metrics across all the implementations mentioned, the respective performances of the two below mentioned algorithms, which use different architectures, have been made comparable to each other. The results obtained by comparing the effectiveness of these algorithms on the same dataset can help gain an insight on the unique attributes of each algorithm, understand how they differ from one another and determine which method of object recognition is most effective for any given scenario.

3. TECHNICAL SPECIFICATION

Design of The Application / Frameworks used

>We have used HTML,CSS and Javascript to create a Basic UI for the User to utilize.
>We have used Node.js to create a backend/ server to fetch user requests.
>We have used the ml5.js libraries to create models for both SSD and YOLO algorithms.
>Canvas Video elements and p5.js is used to perform image processing.
>Pre-trained models for SSD and YOLO are used which can be extended by adding more entries.

In this project we basically have compared two different algorithms of object detection :

- **SINGLE SHOT DETECTOR (SSD) ALGORITHM**

  SSD has two components: a backbone model and SSD head. Backbone model usually is a pre-trained image classification network as a feature extractor. This is typically a network like ResNet trained on ImageNet from which the final fully connected classification layer has been removed. We are thus left with a deep neural network that is able to extract semantic meaning from the input image while preserving the spatial structure of the image albeit at a lower resolution. For ResNet34, the backbone results in a 256 7x7 feature maps for an input image. We will explain what feature and feature map are later on. The SSD head is just one or more convolutional layers added to this backbone and the outputs are interpreted as the bounding boxes and classes of objects in the spatial location of the final layers activations.

  LAYER ARCHITECTURE:

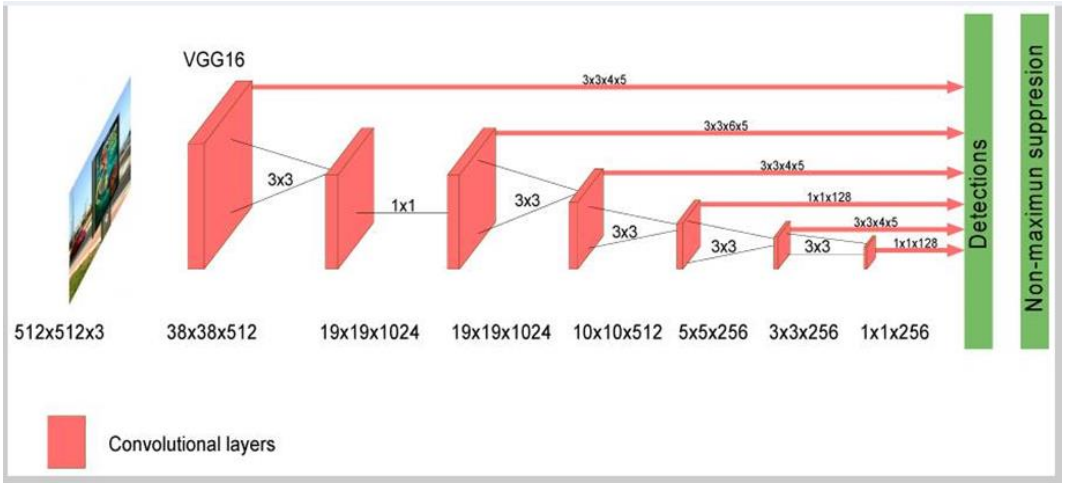

Fig 1 - Layer Architecture of SSD model

FLOWCHART:

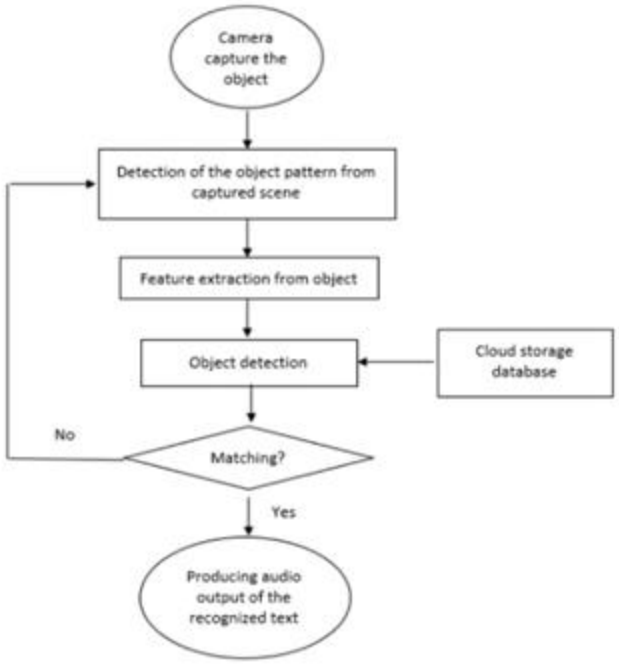

Fig 2 - Flowchart of SSD model

- **YOU ONLY LOOK ONCE (YOLO) ALGORITHM:**

  YOLO was inspired by GoogleNet and the idea was applying a unique neural network to the full image, where the network divides the image into regions and simultaneously predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. YOLO splits an image into a $N \times N$ grid, where each cell predicts only one object. This prediction is given as a fixed number of boundary boxes where each box has its confidence score. It detects one object per grid cell regardless of the number of boxes by applying a non-maxima suppression algorithm. YOLO generally uses ImageNet for parameter pre-training, and then uses target detection data sets for target recognition training. Several improvements on YOLO architecture have been proposed (i.e., YOLOv2 and YOLOv3 versions) which increased the detection accuracy while keeping a very high detection speed.

LAYER ARCHITECTURE:

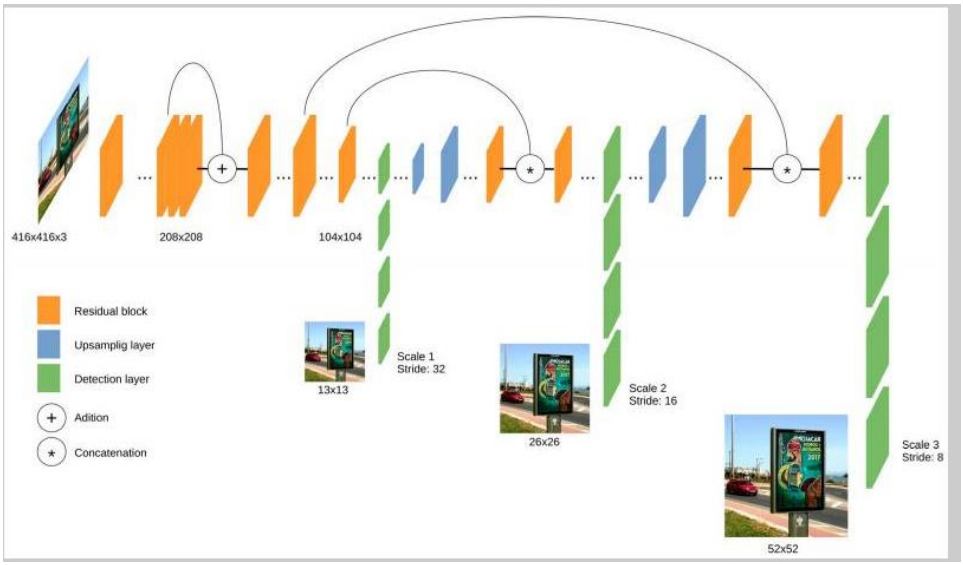

Fig 3 -Layer Architecture of Yolo Model

FLOWCHART:

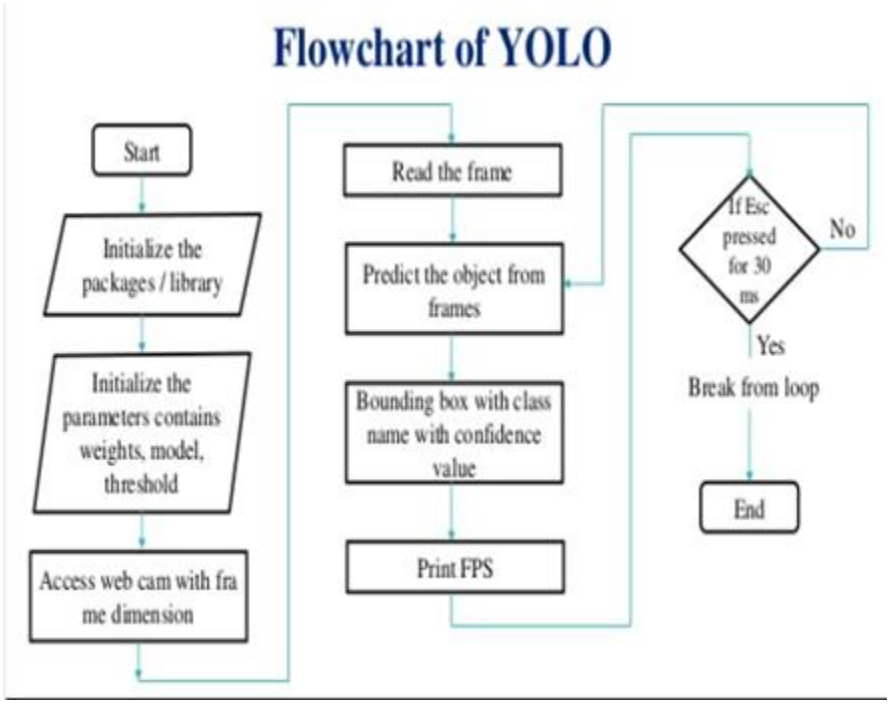

Fig 4 Flow chart of Yolo Model

4. DESIGN APPROACH AND DETAILS

1. Design Approach ,Materials and Method

For Webcam Image Classifier:

➢ The webcam is used to capture a screenshot from video capture.

➢ The received image is flattened to create a 1-D array of pixels.

➢ The flattened array is fed into the YOLO algorithm network.

➢ The YOLO algorithm makes a prediction based on confidence values as to which object is most likely represented by the capture.

➢ The object with the highest confidence value is given as output.

➢ We can employ two different types of algorithm to object detection namely Single shot Detection and YOLO. Single Shot detection uses a Convolution network to perform object detection and thus takes a lot longer to give significant output, since we want out application to be realtime we employed the YOLO algorithm for fast processing.

2. Codes and Standards

1. Main Website Sample Code

index.html

```
<div class="hero">
        <div class="container-fluid">
           <div class="row align-items-center">
              <div class="col-sm-12 col-md-6">
                 <div class="hero-text">
                    <h3>Lets Play with objects..Try out some object Detection</h3>
                    <p>
                       Choose a Model to try
```

```
                    </p>
                    <div class="hero-btn">
                      <a          class="btn"          href="https://jai2901-
webcamimage2.herokuapp.com/">SSD</a>
                      <a          class="btn"          href="https://jai2901-
webcamimage.herokuapp.com/">YOLO</a>
                    </div>
                  </div>
                </div>
                <div class="col-sm-12 col-md-6 d-none d-md-block ">
                  <div class="hero-image">
                    <img
src="https://miro.medium.com/max/3718/1*zcbhlQfmeWNmOlFyaPGPfg.png"
alt="Hero Image">
                  </div>
                </div>
              </div>
            </div>
          </div>
```

index.css

```
body {
  color: #797979;
  background: #ffffff;
  font-family: 'Open Sans', sans-serif;
}

h1,
h2,
h3,
h4,
h5,
h6 {
  color: #343148;
}

a {
  color: #454545;
  transition: .3s;
}

a:hover,
a:active,
```

```
a:focus {
    color: #F7CAC9;
    outline: none;
    text-decoration: none;
}

.btn:focus,
.form-control:focus {
    box-shadow: none;
}

.container-fluid {
    max-width: 1366px;
}

.back-to-top {
    position: fixed;
    display: none;
    background: #2c6c97;
    width: 44px;
    height: 44px;
    text-align: center;
    line-height: 1;
    font-size: 22px;
    right: 15px;
    bottom: 15px;
    transition: background 0.5s;


    z-index: 9;
}

.back-to-top:hover {
    background: #343148;
}
```

2. SSD Sample Code

ssd.html

```
<div id="heading">
        Webcam Image Predictor
    </div>
```

```
<div id="main">

<div>
   <p id="output"></p>
</div>
</div>
<script src="home.js"></script>
```

home.js

```
let classifier;
let video;
function setup(){
   createCanvas(350,350)
   video = createCapture(VIDEO)
   video.size(350,350)
   video.hide()
   classifier = ml5.imageClassifier('MobileNet',video,ReadyModel)}

function ReadyModel(){

     classifier.predict((error,result)=>{
       if(error){
            console.log(error)
            }
            else{
              write = document.getElementById('output')
              write.innerHTML = 'Model Says <b>'+ result[0].label + '</b> With
Confidence <b>' + result[0].confidence*100 + '%</b>'
              ReadyModel()
            }
        })}

function draw(){
image(video,0,0)

}
```

3. YOLO Sample Code

yolo.html

```
<!DOCTYPE html>
<html>
```

```
<head>
  <script
src="https://cdn.jsdelivr.net/npm/p5@1.3.1/lib/p5.min.js"></script>
  <script src="https://unpkg.com/ml5@0.5.0/dist/ml5.min.js"></script>
  <link href="./index2.css" rel="stylesheet">
  <link
href="https://fonts.googleapis.com/css2?family=Montserrat:wght@200&di
splay=swap" rel="stylesheet">
  <meta charset="utf-8" />

</head>

<body>
    <div id="heading">
        Webcam Image Predictor
    </div>
  <script src="home2.js"></script>
</body>

</html>
```

index2.js

```
let video;
let detector;
let detections = {};
let idCount = 0;

function preload() {
  detector = ml5.objectDetector('cocossd');
}

function gotDetections(error, results) {
  if (error) {
    console.error(error);
  }

  let labels = Object.keys(detections);
  for (let label of labels) {
    let objects = detections[label];
    for (let object of objects) {
      object.taken = false;
    }
```

```
}

for (let i = 0; i < results.length; i++) {
  let object = results[i];
  let label = object.label;

  if (detections[label]) {
    let existing = detections[label];
    if (existing.length == 0) {
      object.id = idCount;
      idCount++;
      existing.push(object);
      object.timer = 100;
    } else {
      // Find the object closest?
      let recordDist = Infinity;
      let closest = null;
      for (let candidate of existing) {
        let d = dist(candidate.x, candidate.y, object.x, object.y);
        if (d < recordDist && !candidate.taken) {
          recordDist = d;
          closest = candidate;
        }
      }
      if (closest) {
        // copy x,y,w,h
        let amt = 0.75; //0.75;
        closest.x = lerp(object.x, closest.x, amt);
        closest.y = lerp(object.y, closest.y, amt);
        closest.width = lerp(object.width, closest.width, amt);
        closest.height = lerp(object.height, closest.height, amt);
        closest.taken = true;
        closest.timer = 100;
      } else {
        object.id = idCount;
        idCount++;
        existing.push(object);
        object.timer = 100;
      }
    }
  } else {
    object.id = idCount;
    idCount++;
    detections[label] = [object];
```

```
      object.timer = 100;
    }
  }
  detector.detect(video, gotDetections);
}

function setup() {
  createCanvas(640, 480);
  video = createCapture(VIDEO);
  video.size(640, 480);
  video.hide();
  detector.detect(video, gotDetections);
}

function draw() {
  image(video, 0, 0);

  let labels = Object.keys(detections);
  for (let label of labels) {
    let objects = detections[label];
    for (let i = objects.length - 1; i >= 0; i--) {
      let object = objects[i];
      if (object.label !== "person") {
        stroke(0, 255, 0);
        strokeWeight(4);
        fill(0, 255, 0, object.timer);
        rect(object.x, object.y, object.width, object.height);
        noStroke();
        fill(0);
        textSize(32);
        text(object.label + " " + object.id, object.x + 10, object.y + 24);
      }
      object.timer -= 2;
      if (object.timer < 0) {
        objects.splice(i, 1);
      }
    }
  }
}
```

4. <u>Server Code</u>

app.js

```
const path = require('path');
const express = require('express');
const app = express();
app.use(express.static(path.resolve('./statics/')))
app.use(express.json())
app.use(express.urlencoded({extended:true}))
const port = process.env.PORT || 3000;
app.get('/',function(req,res){
   res.sendFile(path.resolve('./statics/index2.html'));
})
app.listen(port,function(){
   console.log('Running');
})
```

3. Constraints ,Alternatives and Tradeoffs

- 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.
- 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.

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.

Other Alternatives that can be considered are :

- Fast R-CNN - Written in Python and C++ (Caffe), Fast Region-Based Convolutional Network method or Fast R-CNN is a training algorithm for object detection. This algorithm mainly fixes the disadvantages of R-CNN and SPPnet, while improving on their speed and accuracy.

- Faster R-CNN -Faster R-CNN is an object detection algorithm that is similar to R-CNN. This algorithm utilises the Region Proposal Network (RPN) that shares full-image convolutional features with the detection network in a cost-effective manner than R-CNN and Fast R-CNN. A Region Proposal Network is basically a fully convolutional network that simultaneously predicts the object bounds as well as objectness scores at each position of the object and is trained end-to-end to generate high-quality region proposals, which are then used by Fast R-CNN for detection of objects.

- Histogram of Oriented Gradients (HOG)-Histogram of oriented gradients (HOG) is basically a feature descriptor that is utilised to detect objects in image processing and other computer vision techniques. The Histogram of oriented gradients descriptor technique includes occurrences of gradient orientation in localised portions of an image, such as detection window, the region of interest (ROI), among others. One advantage of HOG-like features is their simplicity, and it is easier to understand the information they carry.

- Region-based Convolutional Neural Networks (R-CNN)- The Region-based Convolutional Network method (RCNN) is a combination of region proposals with Convolution Neural Networks (CNNs). R-CNN helps in localising objects with a deep network and training a high-capacity model with only a small quantity of annotated detection data. It achieves excellent object detection accuracy by using a deep ConvNet to classify object proposals. R-CNN has the capability to scale to thousands of object classes without resorting to approximate techniques, including hashing.

- Region-based Fully Convolutional Network (R-FCN) - Region-based Fully Convolutional Networks or R-FCN is a region-based detector for object detection. Unlike other region-based detectors that apply a costly per-region subnetwork such as Fast R-CNN or Faster R-CNN, this region-

based detector is fully convolutional with almost all computation shared on the entire image.

- R-FCN consists of shared, fully convolutional architectures as is the case of FCN that is known to yield a better result than the Faster R-CNN. In this algorithm, all learnable weight layers are convolutional and are designed to classify the ROIs into object categories and backgrounds.

- Spatial Pyramid Pooling (SPP-net)- Spatial Pyramid Pooling (SPP-net) is a network structure that can generate a fixed-length representation regardless of image size/scale. Pyramid pooling is said to be robust to object deformations, and SPP-net improves all CNN-based image classification methods. Using SPP-net, researchers can compute the feature maps from the entire image only once, and then pool features in arbitrary regions (sub-images) to generate fixed-length representations for training the detectors. This method avoids repeatedly computing the convolutional features.

5. SCHEDULE ,TASKS AND MILESTONES

1st : Implementation of SSD
2nd:Implementation of YOLO
3rd: Deploying in web app
4th: Comparison of results

6. PROJECT DEMONSTRATION

Main Website -

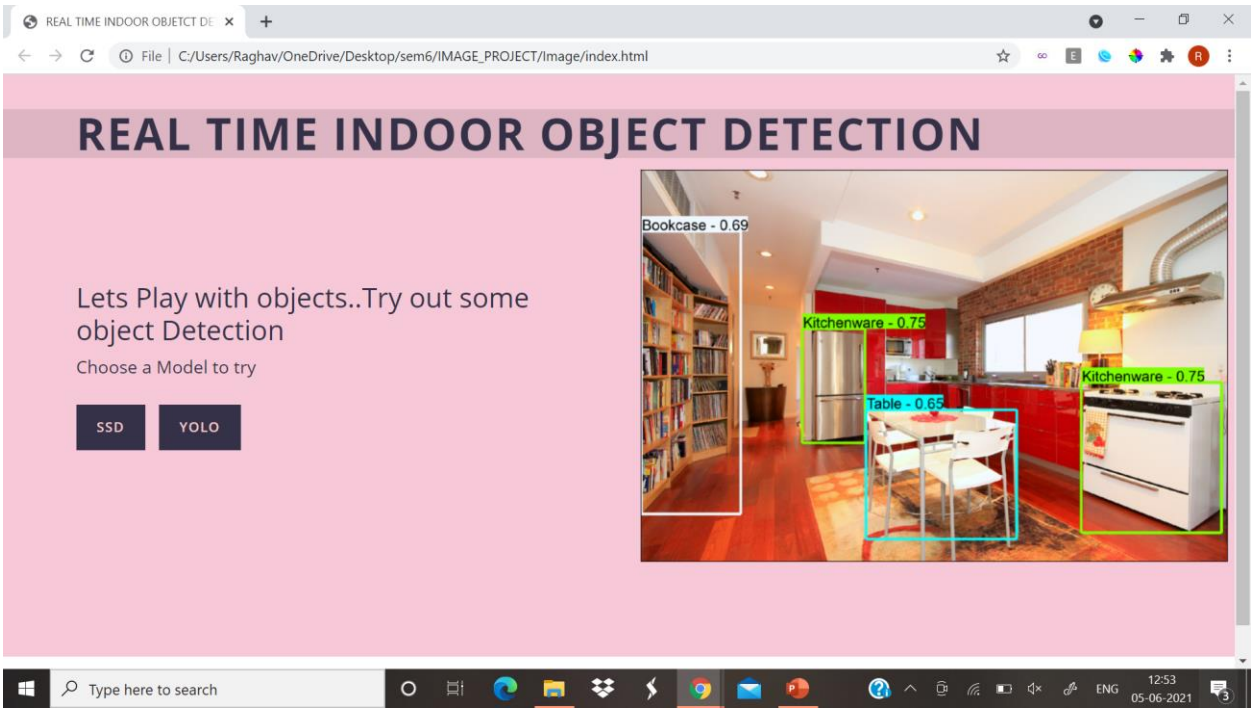

Fig 5 Main Website

Sample SSD Model Outputs

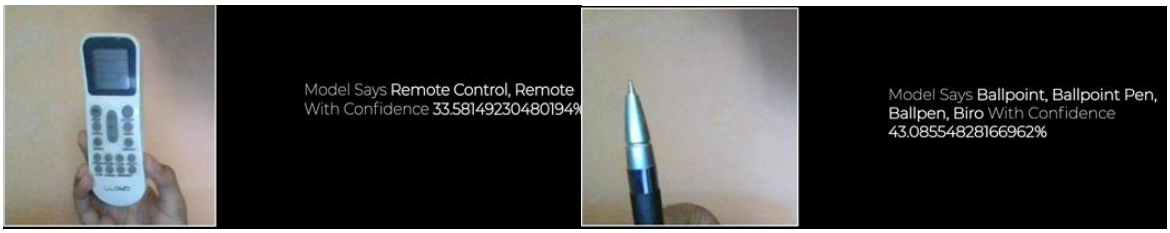

Fig 6 Sample SSD MODEL Outputs

YOLO model Sample Outputs

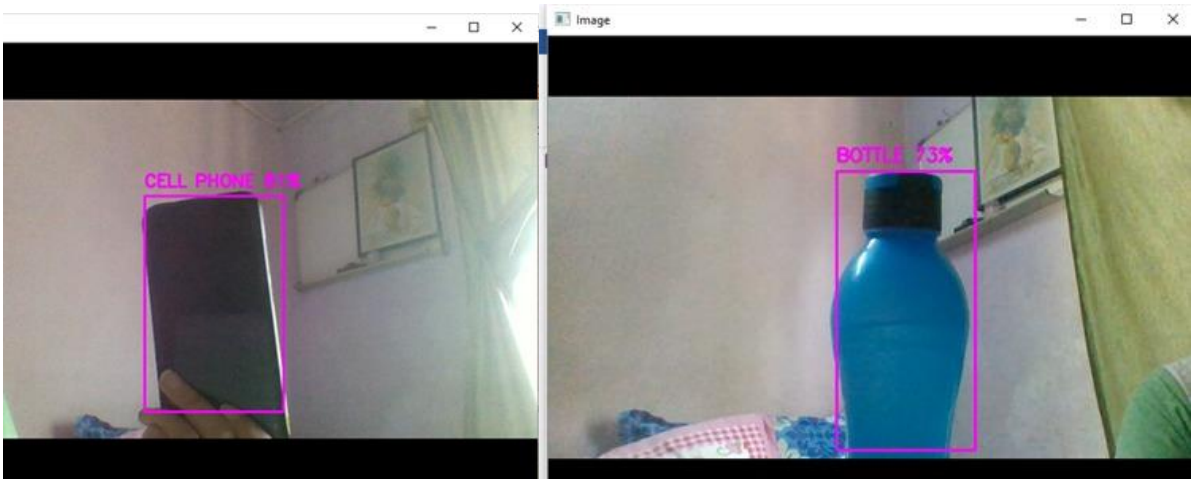

Fig 7 Sample YOLO MODEL Outputs

7. RESULT AND DISCUSSION

Sl.no.	Parameters	YOLO	SSD
1	Model name	You Only Look Once	Single Shot Multi-Box Detector
2	Speed	Low	High
3	Accuracy	80.3% High	72.1% Low
4	Time	0.84~0.9 sec/frame	0.17~0.23 sec/frame
5	Frame per second	45	59
6	Mean Average precision	0.358	0.251

Table 1 Comparison Between SSD and YOLO Model

YOLO (You Only Look Once) system, an open-source method of object detection that can recognize objects in images and videos swiftly whereas SSD (Single Shot Detector) runs a convolutional network on input image only one time and computes a feature map. SSD is a better option as we are able to run it on a video and the exactness trade-off is very modest. SSD is a healthier recommendation. However, if exactness is not too much of disquiet but you want to go super quick, YOLO will be the best way to move forward. First of all, a visual thoughtfulness of swiftness vs precision trade-off would differentiate them well. While dealing with large sizes, SSD

seems to perform well, but when we look at the accurateness numbers when the object size is small, the performance dips a bit.

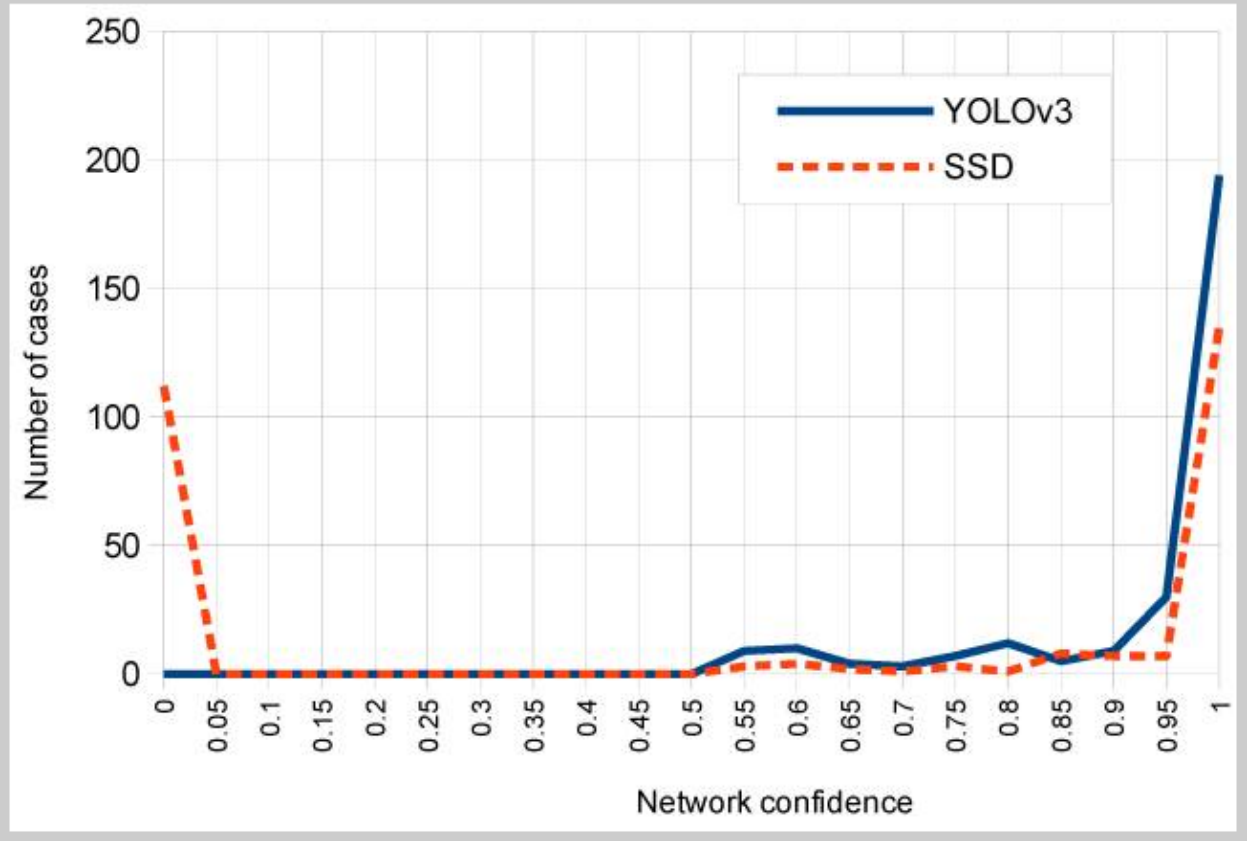

Fig 8 Comparison between Yolo and SSD model

Note that for YOLOv3 most cases correspond to a high confidence value between 0.95 and 1, while for SSD the most returned confidence values on detections are distributed in ranges between 0.95 and 1 (first place) and between 0 and 0.05 (second place).