*Dissertation on*

# "Image Annotation Based Information retrieval"

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

**Bachelor of Technology**
**in**
**Computer Science & Engineering**

**UE17CS490A – Capstone Project Phase - 1**

***Submitted by:***

| | |
|---|---|
| **Jnanesh** | **PES1201701822** |
| **Sachin** | **PES1201701725** |

*Under the guidance of*

**Prof. Surabhi Narayana**
PES University

**August - December 2020**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
FACULTY OF ENGINEERING
**PES UNIVERSITY**
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

**PES UNIVERSITY**

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

**FACULTY OF ENGINEERING**

# CERTIFICATE

*This is to certify that the dissertation entitled*

**'Image Annotation based Information Retrieval'**

*is a Bonafede work carried out by*

| | |
|---|---|
| **Jnanesh** | **PES1201701822** |
| **Sachin** | **PES1201701725** |

in partial fulfilment for the completion of eighth semester Capstone Project Phase - 2 (UE17CS490B) in the Program of Study - Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period Jan. 2021 – May. 2021. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 8th semester academic requirements in respect of project work.

| | | |
|---|---|---|
| Signature | Signature | Signature |
| **Surabhi Narayana** | Dr. Shylaja S S | Dr. B K Keshavan |
| Prof. | Chairperson | Dean of Faculty |

**External Viva**

| **Name of the Examiners** | **Signature with Date** |
|---|---|
| 1.________________________ | ________________________ |
| 2.________________________ | ________________________ |

## DECLARATION

We hereby declare that the Capstone Project Phase - 1 entitled **“Image Annotation Based Information Retrieval”** has been carried out by us under the guidance of Prof. Surabhi Narayana and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester August – December 2020. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

| | | |
|---|---|---|
| **PES1201701822** | **Jnanesh** | Jnanesh D |
| **PES1201701725** | **Sachin** | Sachin. |

# ACKNOWLEDGEMENT

I would like to express my gratitude to Prof. Surabhi Narayan, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE17CS490A - Capstone Project Phase – 1.

I am grateful to the project coordinators, Prof. Spurthi N Anjan, and Sylvia Nancy J for organizing, managing, and helping with the entire process.

I take this opportunity to thank Dr. Shylaja S S, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department. I would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this project could not have been completed without the continual support and encouragement I have received from my family and friends.

# ABSTRACT

Object Recognition is a challenging and exciting task of computer Vision. Object recognition is to describe a collection of related computer vision tasks which involves identifying objects in images. Image classification involves predicting the class of objects in an image. Object localization refers to identifying the location of objects in an image. Image Annotation in Machine Learning is the process of creating bounding boxes around the localized images. Now with the advance of deep learning and neural networks, we can finally tackle these problems without coming up with various heuristics real time. By now, there are many different algorithms and concepts for object detection published. Just a few are lightweight enough to run them on live interface that results in an appropriate frame rate. Object detection networks are for example the You Look Only Once (in short: YOLO) network or the Faster R- CNN network.

In this project our aim was to develop a software application that takes an image and produce the Informative links about the objects present in the image. Our aim in this project is to develop a software product that runs the SSD to identify objects in an image and the identified images are searched in Google API to give appropriate informative links of those objects.

# TABLE OF CONTENT

| Chapter No. | Title | Page No. |
|---|---|---|
| **1.** | **INTRODUCTION** | 2 |
| **2.** | **PROBLEM STATEMENT** | 4 |
| **3.** | **LITERATURE REVIEW** | 6 |
| | 3.1 Structural Inference Network | 6 |
| | 3.2 Single Shot MultiBox Detector. | 7 |
| | 3.3 You Only Look Once: Unified Real Time object detection | 10 |
| | 3.4 Faster R-CNN: Towards Real Time object Detection with Regional Proposal Network. | 11 |
| | 3.5 Very Deep Convolutional Network | 12 |
| | 3.6 Google Vision API | 13 |
| **4.** | **DATASET** | 15 |
| | 4.1 Overview | 15 |
| | 4.2 Features | 15 |
| | 4.3 Main use of Data | 15 |
| | 4.4 Basic Attributes of COCO | 17 |
| | 4.4.1 Structure | 17 |
| | 4.4.2 Licenses | 18 |
| | 4.4.3 Images | 18 |
| | 4.4.4 Categories | 19 |
| | 4.4.5 Annotations | 21 |
| **5.** | **PROJECT REQUIREMENTS SPECIFICATION** | 22 |
| | 5.1 Introduction | 22 |
| | 5.2 General description | 23 |
| | 5.3 Functional Requirements | 23 |
| | 5.4 Interface Requirements | 24 |
| | 5.5 Performance Requirements | 24 |
| | 5.6 Design Constrains | 25 |
| **6.** | **SYSTEM REQUIREMENTS SPECIFICATION** | 26 |
| | 6.1 Hardware Requirements | 26 |
| | 6.2 Software Requirements | 26 |

**7.** **SYSTEM DESIGN** 27

**8.** **LOW LEVEL DESIGN** 29

8.1 Object Detection Model Deployment 29

8.2 Working of Information Server 31

8.3 Request Handling by Object Detection Server 35

8.4 Sequence Diagram 36

8.5 Use Case Diagram 37

8.6 Information REST API Server Diagram 38

8.7 Object Detection Server Diagram 39

8.8 Class Diagram of SSD Model 40

**9.** **USED GUIDE** 41

**10.** **EVALUATION OF SSD MODEL** 46

**11.** **CONCLUSION AND FUTURE WORK** 47

**REFERENCE/ BIBLIOGRAPHY** 49

**APPENDIX A DEFINITIONS AND ABBREVIATIONS**

## LIST OF FIGURES

| Figure No. | Title | Page No. |
|---|---|---|
| Figure 1 | SSD Model | 8 |
| Figure 2 | SSD Object Detection Example | 16 |
| Figure 3 | COCO Dataset Images | 16 |
| Figure 4 | COCO Class with Image Count | 20 |
| Figure 5 | System Design Diagram | 27 |
| Figure 6 | Sequence Diagram | 36 |
| Figure 7 | Use Case Diagram | 37 |
| Figure 8 | Information REST API Server Diagram | 38 |
| Figure 9 | Object Detection Server Diagram | 39 |
| Figure 10 | Class Diagram of SSD Model | 40 |
| Figure 11 | UI Input Options | 41 |
| Figure 12 | Cropping and Detect Requestion in UI | 42 |
| Figure 13 | Waiting for Response in UI | 43 |
| Figure 14 | Information Server Request info UI | 44 |
| Figure 15 | System Response Display UI | 45 |

## LIST OF TABLES

| Table No. | Title | Page No. |
|---|---|---|
| Table 1 | Object Detection Model Comparison | 7 |
| Table 2 | COCO Classes | 19 |
| Table 3 | Evaluation of SSD | 46 |

# CHAPTER 1

# INTRODUCTION

Vision and perception come naturally to us humans. We never think how hard visual intelligence can be. It was up until the mid-20th century that vision and perception was not considered to be an intelligent task. Vision and perception were considered an easy approach as compared to reasoning and planning. Back then artificial intelligence was mostly a theory. With further advancements in artificial intelligence, when attempts were made, in the mid- 20th century to build AI with reasoning and planning and AI with vision and perception, it was realized that vision and perception can be often harder than reasoning (may be not planning!).

We have a very limited understanding of human/animal visual perception abilities. It is so, we believe artificial visual intelligence is very hard. A more in depth understanding of the human/visual perception system may help us build more efficient systems with much more ease. ANN was an approach in this direction. Invented by Geoffrey Hinton it was built to loosely mimic the human brain. ANN was a very successful approach in artificial intelligence. With backpropagation (learning, error and correction method), ANN produced very impressive results in every aspect of artificial intelligence. However, it was not considered a correct path to artificial general intelligence. The primary reason was hardware limitations. It was

In the mid-2000s the problem of hardware was overcome with very powerful GPUs. Since then we have seen the fastest growth in the field of AI. CNN was invented by the student of Geoffrey Hinton, Yann Lecun. But, can artificial general intelligence be solved with neural networks? Maybe, but we still do not understand a lot about intelligence and intelligent models, so maybe not. It is very difficult to answer this question precisely. So why this approach of neural networks? Because this produces results. We have seen results with CNN that were not possible to imagine a decade ago. Facial recognition, image classification, Automatic Image Annotation a single model to classify and label up to 1000, synthetic data generation with GANs. As of now we believe this a direction of study which needs to be extensively explored.

Our B.Tech. project and report is one such attempt to explore this vast and extensively growing field of computational intelligence. We had some prior experience in this field and wanted to explore this even further. We primarily studied automatic image annotation and object detection. We present our findings and results in this report.

# CHAPTER 2

# PROBLEM STATEMENT

Computer Vision is a field of Computer Science that deals with the question of how a machine can be made to understand a scene (a digital image). By understanding it meant to extract various meaningful features from the image. These meaningful features can be used to extract more complex objects, like car, dog, cat etc. The purpose of Computer Vision is to automate the work human visual systems can perform and more. Automatic Image Annotation is a computer technology that deals with the recognition and localization of the object class in an image. Every class has its own special features that helps in classifying the class for example all circles are round. Object Detection uses these special features.

In this project we present an Object Detector Single Shot MultiBox Detector (in short: SSD) which delivers great precision while being faster than other neural networks. Let's take a look at the components in the name:

- Single Shot: this means that the tasks of object classification and localization are done in a single feedforward network.
- MultiBox: this is the name of a technique for bounding box regression developed by *Szegedy*.
- Detector: the network is an object detector that also classifies those detected objects.

The detected objects are cropped and searched in google to find the information about the objects and links of those will be made accordingly to the end user. Google Image search is the largest and most popular image search as it offers simple and advanced search which is quick and easy.

By utilizing this on the segmented objects we give the user about the information of objects which are in the image through a web interface.
In the first phase of the Capstone Project we train the SSD model based on the Pytorch Machine Learning Library on Dataset called COCO (Common Objects in Context). We take an input image of 300 x 300 for the base network VGG 16 and identify the objects in the image through bounding box annotation by SSD model based on the pytorch library.

The main objective of our project is to give the appropriate information about the objects in an image they are having. Multiple objects of different classes present in an image are identified through the SSD model and are searched individually in the google to get the information about them and presently in a user-friendly manner on a web - interface.

SSD model can identify the object classes of about 80 classes of COCO Dataset. It provides a better mAP without significant decrease in speed, so that object detection can be carried out in real time with a considerable speed

# CHAPTER 3

# LITERATURE SURVEY

In this chapter, we present the various published research papers on Computer Vision and object detection

## 3.1 Structural Inference Network: Object Detection Using Scene Level Context and Instance Relationship - Published in CVPR 2018

In this paper, the author proposes an object detection algorithm which takes into account not only the visual appearances of objects, and the use of object relation scenes and contextual information in a single image.

It is considered as a cognitive as well as a reasoning problem. Structural inference network incorporates a typical framework with a graph model which aims to infer the object state.

Architecture: The base detection framework used is Faster R-CNN. GRU as memory cell is presented in this work. SIN presents a graph to model the graphical problem.

Results:

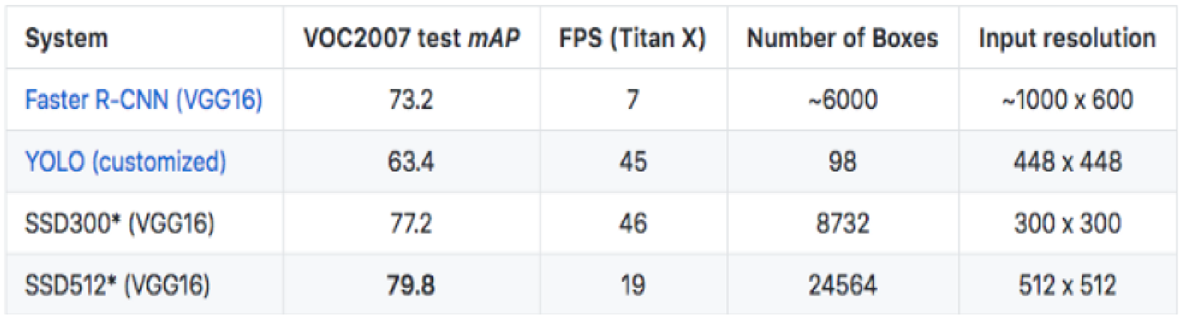
- By using base network as Faster R-CNN model the SIN has achieved a mAP of 73.1 when trained on PASCAL VOC 2007+12
- Scene level contextual information has helped to achieve significant higher accuracy.
- Slower than Faster R-CNN due to GRU implementation of object scene relationship

## 3.2 Single Shot MultiBox Detector. Wei Liu, Dragomir Angelov, Dumitri Erhan, Christian Szegedy, Scott Red, Cheng-yang fu, Alexander C Berg

SSD is an Object Detection model which was built as a deep neural network. This model takes one single shot to detect multiple objects in an image. Single shot model condenses both localization and detection responsibilities in a single forward sweep of the network, resulting in significantly quicker detection while it can be deployed on lighter common usage hardware. Multibox is technique that predicts an objects bounding boxes as a regression problem.

The model takes a set of default bounding boxes over dissimilar aspect fractions and scales per feature-map location and gradient descent to optimize. During prediction the network generates probability for each object category present in each default box. The network combines predictions from multiple feature maps with different resolutions through IoU.

SSD is designed to detect the items in real time. Faster R-CNN uses a Regional Proposal Network to generate bounding boxes around items to identify them, it may be considered very accurate but the whole procedure routine is just 7 frames per second. SSD speeds up the process by eliminating need for RPN. To recover the precision it applies developments including the multi-scale features and default boxes.

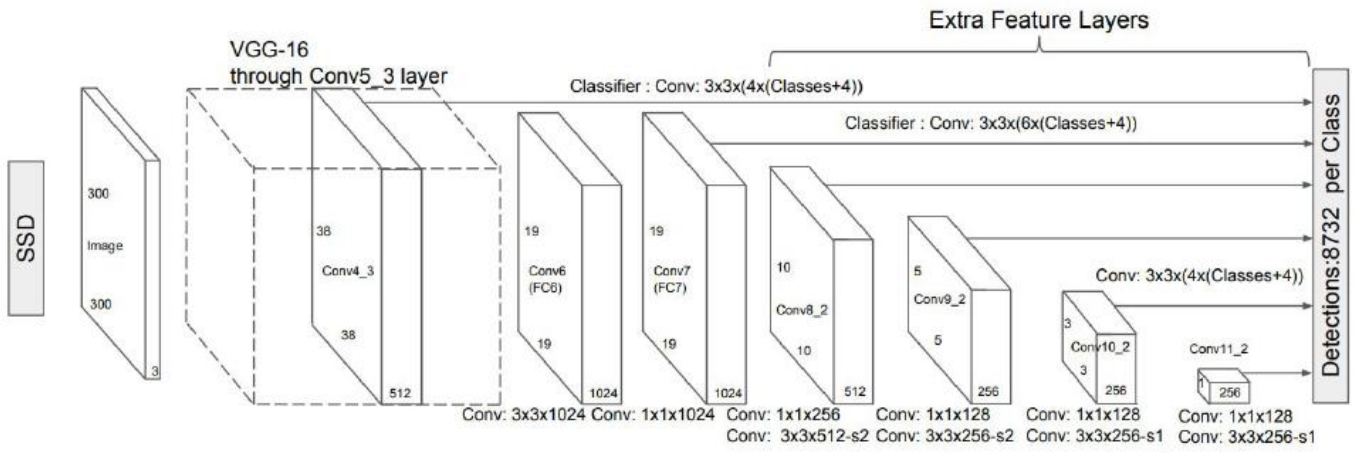


| System | VOC2007 test *mAP* | FPS (Titan X) | Number of Boxes | Input resolution |
|---|---|---|---|---|
| Faster R-CNN (VGG16) | 73.2 | 7 | ~6000 | ~1000 x 600 |
| YOLO (customized) | 63.4 | 45 | 98 | 448 x 448 |
| SSD300* (VGG16) | 77.2 | 46 | 8732 | 300 x 300 |
| SSD512* (VGG16) | **79.8** | 19 | 24564 | 512 x 512 |

Architecture: SSD is built on the VGG-16 but discards the fully connected layers. The features maps obtained by VGG is used to detect the objects through regression. A set of auxiliary convolutional layers were added, thus enabling to extract features at multiple scales and multiple aspect ratios and progressively decreasing the size of the input to each subsequent layer. Each prediction of SSD comes in boundary box and 81 scores for each class of the object (one extra for background), and we have to filter out based on the object threshold. Conv4_3 is the filter that is used to apply on the feature maps and it makes a total of 38*38*4 predictions.

SSD uses multi scale feature maps to detect the items independently. The CNN has reduced the spatial dimension gradually, the resolution of the feature maps also gets decreased. To detect the larger items SSD uses lower resolution layers.

Also, SSD adds more supplementary convolutional layers after the VGG16. Five of them will be used to detect the objects.



Each image is divided into multiple cells and the SSD applies the convolutional filters for each cell. Each filter produces 85 channels: 81 classes and one boundary box.

Advantages:

- SSD 300 model achieves best mAP on PASCAL VOC 2007 at 74.3 at 59fps.
- Default boxes in SSD results in more accurate detection.
- Accuracy rises with the number of default boundary boxes at the cost of detection speed.
- COCO dataset has small sized objects and to improve the accuracy of the detection we use smaller default boxes.
- SSD also has minor localization faults compared to the R-CNN but slightly more classification error when compared to the R-CNN.

## Conclusion

SSD single shot multi box detector. It has no surrogate regional proposal network to detect the regions of interest. As an alternative, it forecasts the bounding boxes and the classes directly from the feature maps in one pass.

To improve the accuracy of the model.

- Minor convolutional filters are used to guess the object classes and offsets to default bounding boxes.
- Separate filters for each default bounding box to handle different aspect ratios.
- Multi scale feature maps used for the object detection.

SSD can be accomplished end-end for improved accuracy. SSD makes more forecasts and has better coverage on the item location, scale and aspect ratios. With the enhancements above, SSD can reduce the input image resolution to 300*300. The model can run at the real time speed and still beats the accuracy of the state-of-the-art model Faster-RCNN.

## 3.3 You Only Look Once: Unified Real Time object detection, Joseph Redmon, Santhosh Divvalay, Ross Girshick, Ali Farhadiy 2016

YOLO is an approach to object detection problem. It frames object detection as a bounding box regressor problem and associates the class probability directly from full images in one evaluation.

A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation through the network.

YOLO divides an image into SxS grid. Each grid cell predicts a fixed number of bounding boxes to locate the objects.

Advantages of using YOLO:

- Fast. Good for real time processing.
- Predictions (object locations and classes) are made from single network.
- Yolo is more generalized. It outperforms other methods when generalizing from natural images to other domains.
- Achieves significant speed of up to 155 fps

## 3.4 Faster R-CNN: Towards Real Time object Detection with Regional Proposal Network. Shaoquin Ren Kaiming, Ross Girschick, Jian Sun - 2016

Faster R-CNN is the most widely used version of R-CNN family of object detection models.

A Faster R-CNN object detection network is divided into two parts. The first part of faster R-CNN is regional proposal network which mainly focus on creating weighted region of interest on an image which has high probability of existence of an object and the second part is prediction of object class. In this part a region of interest is given to a CNN network which predicts the object class and CNN is trained to produce region proposal without any selective3 search mechanism.

Results:

- Detection with a VGG RPN takes 198ms compared to the 1.8 seconds of Selective Search of Older methods.
- It has the accuracy of 75.9 when trained on COCO and PASCAL VOC Datasets
- Faster RCNN using Inception Resnet with 300 proposals gives the highest accuracy at 1 FPS for all the tested cases

## 3.5 VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE SCALE IMAGE RECOGNITION. Karen Simonyan and Andrew Zisserman 2015

VGG is an object-recognition model which mainly focuses on the image classification. VGG outperforms on many tasks and datasets outside of ImageNet. This model is now still one of the most used image-recognition model. The design opted by the authors are similar to Alexnet which is increase the number of feature maps or convolution as the depth of the network increases.

### Architecture:

VGG mainly focussed on classification on image classification accurately.

VGG takes an image of size 224x224. The convolutional layers in VGG has a very small receptive field of 3*3. VGG has three fully connected layers. All of the VGG's hidden layers use a ReLU. VGG incorporates 1x1 convolutional layers for making decision functions more nonlinear without changing receptive fields.

### Results:

VGG has a top1 error of 25.5% and top5 error of 8.5%. It has achieved state of the art performance on the ImageNet challenge.

## 3.6 A PRACTICAL STUDY ABOUT THE GOOGLE VISION API *October 6th 2016. Daniel Pedro Ferreira Lopes, Antonio J. R*

Google Vision API is an API provided by the google which allows the developers to understand the image content. Due to availability of many machine learning models in background it becomes most powerful and useful tool to developers. There are many examples presented to practical study about the usage of google vision library in order to study google vision features to show usability in real world applications.

Nowadays life has thousands of problems which could be

solved through computer vision solutions and there are many projects under development all over the world. Problems like emotional states, OCR, face recognition, Object detection and image classification can be solved with these solutions which are available in technology. Google vision API is low cost solutions that currently exists in the world.

Google Vision API provides many features like:

I. Label Detection
II. Face detection
III. Landmark detection
IV. OCR (Optical character recognition)
V. Explicit content Detection
VI. Logo Detection

Label Detection is a feature that detects the various set of categories and classes of image belongs to ranging from objects, animals to popular places. The response given to us with a confidence score (between 0 and 1)

Landmark Detection detects popular places which are natural and man-made which is in the image. Landmark detection algorithm is very precise. From the images, it either gives us the exact response or does not give anything at all.

Face Detection is a feature that detects the maximum faces in an image, along with the various key facial attributes like wearing head ware or emotional state. The potential of google to detect the emotion of the human face is more accurate. Face detection algorithm that the google uses is very powerful. It was tested with numerous images and google API detected all of them.

## Final Results:

After this study it is undoubtful that Google's Vision API has a great potential and it is always getting better with time. Its powerful ML algorithms and large database is helping it to be best market solutions.

Landmark detection has a good feature in labelling images that were taken during a function, trips and organized them in albums. And face detection, it may help the future research on people's emotions identification when exposed to a certain environment.

# CHAPTER 4

# DATASET

## 4.1 Overview

Here we discuss the dataset used in this project and its features. COCO is a large scale, high quality datasets for object detection and computer vision tasks.

## 4.2 Features

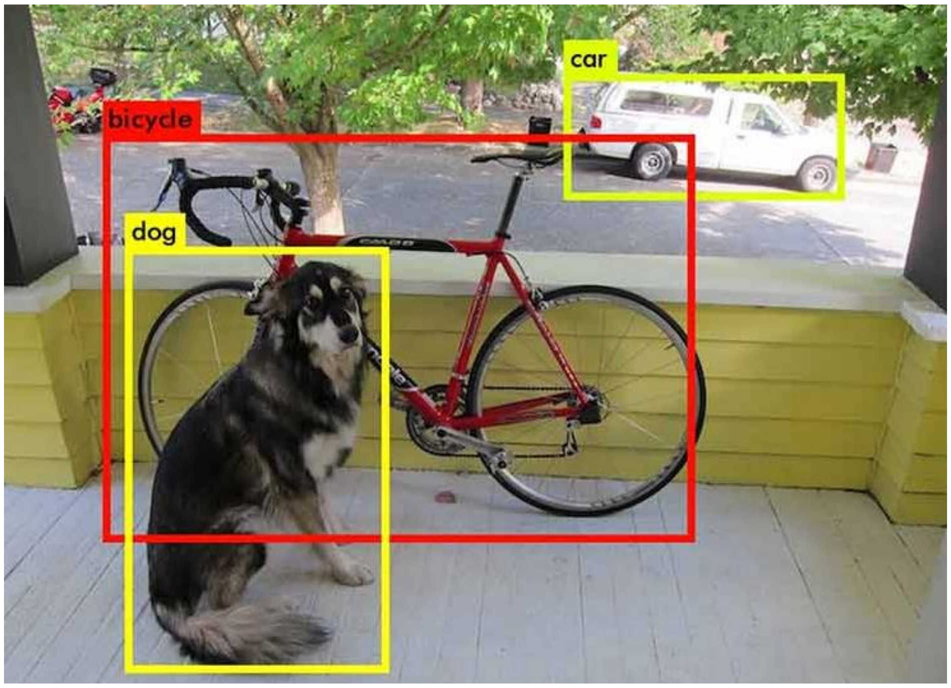
COCO Dataset features:

- Object Segmentation
- Image Annotation
- Recognition in Context
- Stuff Segmentation
- It has more than 330,000 images with more than 200000 images labelled
- It has 1.5 million object instances
- It has 80 different object classes
- There are even tools built specifically to work with datasets in COCO format, e.g., COCO annotator, COCO API's.

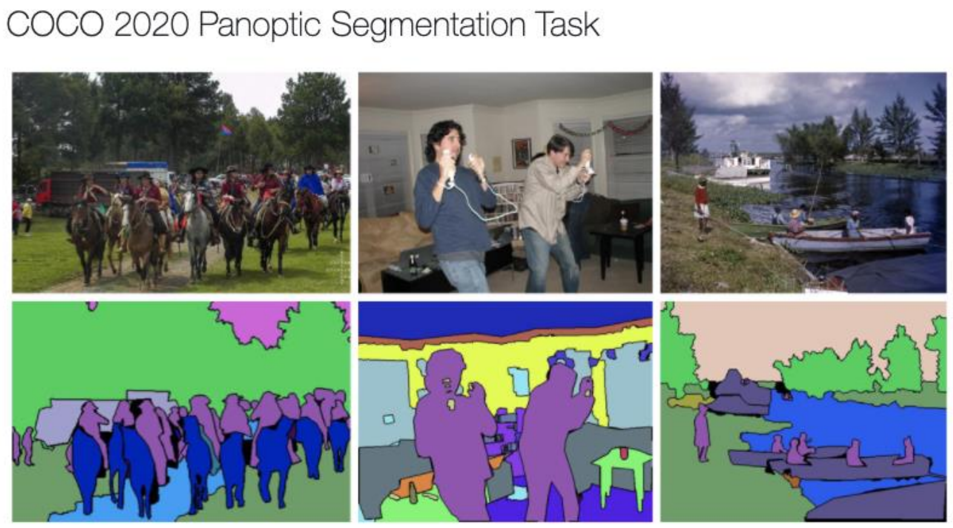
## 4.3 Main uses of Data

Coco dataset is mainly used for three important tasks.

1. Object Detection - Identification of objects in an image through bounding boxes that it list of object classes and coordinates of rectangles around them; objects are discrete, separate objects, often with parts, like humans and vehicles; the official dataset for this task also contains additional data for segmentation



2. Object segmentation – Polygonal segmentation of objects in an image.



3. Stuff segmentation – Segmentation of objects as well as background using continuous patterns.

In computer vision, these tasks have wide applications in self driving cars and other computer vision tasks.

## 4.4 Basic Structure of COCO Dataset

Format of coco annotations is JSON, which has key value pairs inside braces, as a top value. It has additional lists and dictionaries inside the value.

### 4.4.1 Its Structure is as follows:

```
{
“Info”:””,
“Licenses”:””,
“Images”:””,
“Categories”:””,
“Annotations”:” {}”
}
```

Info Section: It contains the metadata about the dataset. It contains a url pointing to the dataset website, year, version, date created and so on.

4.4.2 Licenses: Here will be the links for licenses for images in the dataset with structure as

```
{
“url”:””,
“Id”:”1”,
“name”:”Attribution-NonCommercial-ShareAlike License”
}
```

4.4.3 Images: This part of file contains the metadata about the images.

```
{
"license": 3,
"file_name": "0000008921895.jpg",
"coco_url": "http://images.cocodataset.org/train2017/000000391895.jpg",
"height": 360,
"width": 640,
"date_captured": "2013–11–14 11:18:45",
“flickr_url”:”www.imageurl.com/391895.jpg”
}
```

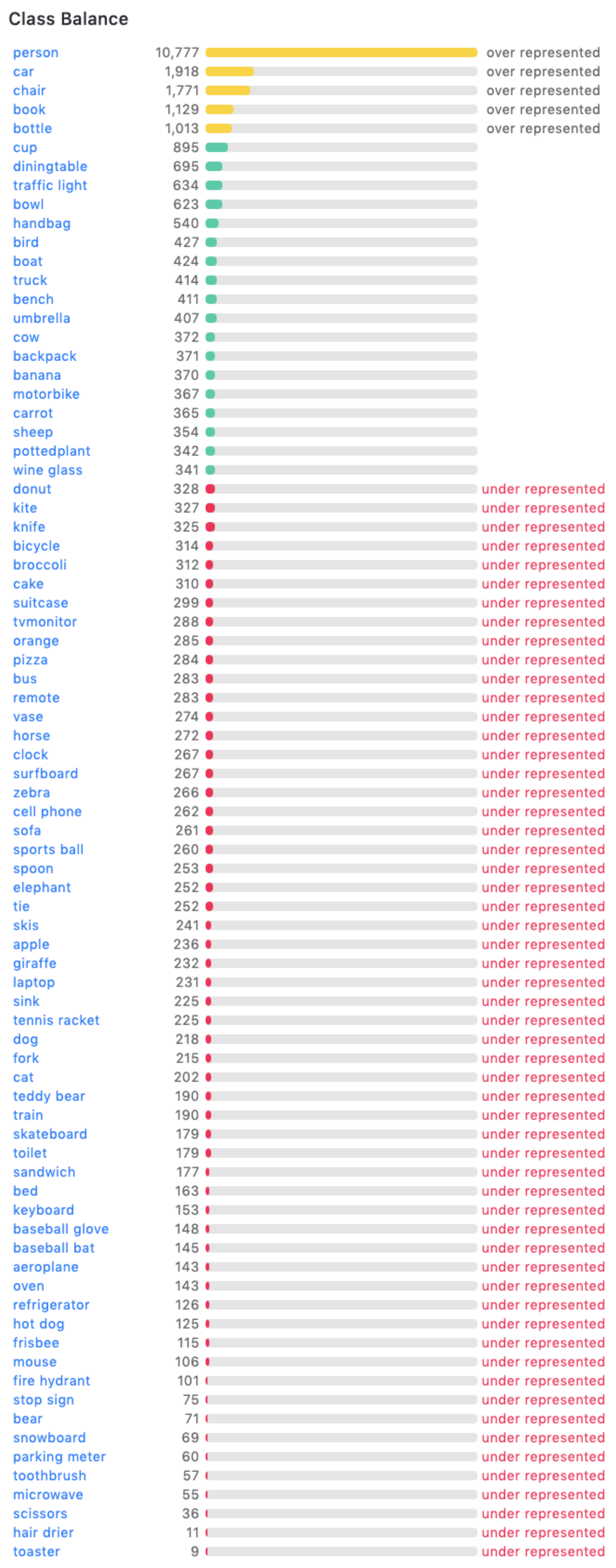
Through id field we can get the annotations of an image through the id field of the images in the annotations file by searching id.

4.4.5 Categories:

There are 80 classes of objects that can be identified on images in COCO. Each category has a unique id. Also contains a super category which groups the related categories.

Classes:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Person | Car | Bike | Chair | Book | Bottle | Cup | Dining Table |
| Traffic Light | Bowl | Hand bag | Bird | Boat | Truck | Bench | Umbrell a |
| Cow | Backpa ck | Banana | Motor Bike | Carrot | Sheep | Potted plant | Wine Glass |
| Donut | Kite | Bicycle | Broccol i | Cake | Suit case | Tv Monitor | Orange |
| Pizza | Bus | Remote | Vase | Horse | Clock | Surf Board | Zebra |
| Cell Phone | Sofa | Sports Ball | Spoon | Elephant | Tie | Skis | Apple |
| Giraffe | Laptop | Sink | Tennis Racket | Dog | Fork | Cat | Teddy bear |
| Train | Skate Board | Toilet | Sand wich | Bed | Key board | baseball glove | Baseball bat |
| Aeropl ane | Oven | Refrige-rator | Hotdog | Frisbee | Mouse | Fire Hydrant | Stop Sign |
| Bear | Snow Board | Parking Meter | Tooth Brush | Micro wave | Scisso rs | Hair Dryer | Toaster |



Class Balance

| | | |
|---|---|---|
| person | 10,777 | over represented |
| car | 1,918 | over represented |
| chair | 1,771 | over represented |
| book | 1,129 | over represented |
| bottle | 1,013 | over represented |
| cup | 895 | |
| diningtable | 695 | |
| traffic light | 634 | |
| bowl | 623 | |
| handbag | 540 | |
| bird | 427 | |
| boat | 424 | |
| truck | 414 | |
| bench | 411 | |
| umbrella | 407 | |
| cow | 372 | |
| backpack | 371 | |
| banana | 370 | |
| motorbike | 367 | |
| carrot | 365 | |
| sheep | 354 | |
| pottedplant | 342 | |
| wine glass | 341 | |
| donut | 328 | under represented |
| kite | 327 | under represented |
| knife | 325 | under represented |
| bicycle | 314 | under represented |
| broccoli | 312 | under represented |
| cake | 310 | under represented |
| suitcase | 299 | under represented |
| tvmonitor | 288 | under represented |
| orange | 285 | under represented |
| pizza | 284 | under represented |
| bus | 283 | under represented |
| remote | 283 | under represented |
| vase | 274 | under represented |
| horse | 272 | under represented |
| clock | 267 | under represented |
| surfboard | 267 | under represented |
| zebra | 266 | under represented |
| cell phone | 262 | under represented |
| sofa | 261 | under represented |
| sports ball | 260 | under represented |
| spoon | 253 | under represented |
| elephant | 252 | under represented |
| tie | 252 | under represented |
| skis | 241 | under represented |
| apple | 236 | under represented |
| giraffe | 232 | under represented |
| laptop | 231 | under represented |
| sink | 225 | under represented |
| tennis racket | 225 | under represented |
| dog | 218 | under represented |
| fork | 215 | under represented |
| cat | 202 | under represented |
| teddy bear | 190 | under represented |
| train | 190 | under represented |
| skateboard | 179 | under represented |
| toilet | 179 | under represented |
| sandwich | 177 | under represented |
| bed | 163 | under represented |
| keyboard | 153 | under represented |
| baseball glove | 148 | under represented |
| baseball bat | 145 | under represented |
| aeroplane | 143 | under represented |
| oven | 143 | under represented |
| refrigerator | 126 | under represented |
| hot dog | 125 | under represented |
| frisbee | 115 | under represented |
| mouse | 106 | under represented |
| fire hydrant | 101 | under represented |
| stop sign | 75 | under represented |
| bear | 71 | under represented |
| snowboard | 69 | under represented |
| parking meter | 60 | under represented |
| toothbrush | 57 | under represented |
| microwave | 55 | under represented |
| scissors | 36 | under represented |
| hair drier | 11 | under represented |
| toaster | 9 | under represented |

### 4.4.6 Annotations:

The annotations are the important feature of this dataset. It contains information about

the different types of annotations about the objects in the images.

It contains a dictionary structure for each object as follows:

```
{
        “Segmentations”:””,
        “Area”:””,
        “isCrows”:””,
        “Bbox”:””,
        “Image_id”:””,
        “Category_id”:””,
        “Id”:””,
}
```

Segmentation : List of segmentation mask pixels.

Area : Number of pixels inside the annotaion

isCrowd : 0 for single object and 1 for multiple close objects.

Image_id : The id field of images dictionary

Bbox - The bounding box parameters Category_id - class of the object

Id - Unique identifier for each object

# CHAPTER 5

# PROJECT REQUIREMENTS SPECIFICATION

Computer vision is a field of computer science where computer visualizes or identifies the real-world entities. It will be helpful in general image understanding in tasks such as Content based image retrieval. Our project mainly focuses on annotation of multiple objects in a given image and generating the informative links which are useful for the users.

## 5.1 Introduction:

- **Purpose of this Document –**

  This document mainly describes project requirements like general description, functional requirements, interface requirements and performance requirements. Gives proper ideas about all requirements.
- **Scope of this document –**

  The overall working and main objective of the document and what value it will provide to customers is described and explained.
- **Overview**

  Our product takes in the image as input. By using the SDD model it generates the rectangular box on each object in an image. And then by using OpenCV and Google API's our product generates the informative links which are useful to users.

## 5.2 General description:

Product model is developed by training the CNN on COCO dataset. And by using web frameworks, user interface has to be developed. Such that users can upload the image from which they want to retrieve information.

Just by uploading an image to our model they get all objects annotated with some important link in a minimum time span of around half a second. So, this simplifies the image identification and getting information for users.

## 5.3 Functional Requirements:

- For building Image Annotation model following functions are done
- Initially the dataset needs to be downloaded.
- Installing all the packages of particular versions which support Models like Py-Torch 0.2, OpenCV etc.
- Construction of Convolutional neural network normally vgg16 model is used as the base model for construction.
- Setting up all configurations to the model, adding learning rate, loss function and optimizer etc.
- Then training the model on a particular set of datasets.
- Followed by testing the system and decreasing the loss function value on each iteration.
- Then finally running on the validation set of the dataset for checking the accuracy of the model.
- Then collecting the weights of the model. Model training finishes here.

- Users can directly interact with the model with help of provided user interface.

- Users can directly upload images to the model. Using a trained weights model detects objects very fast.

- Gives output as bounding box on each object with annotation and coordinates of bounding box. Using Open-cv we crop the image using bounding box coordinates.
- And search these images in google using google images api and give useful information to users.

**5.4 Interface Requirements:**

- User interface is developed using web frameworks like HTML, CSS, JavaScript, Node-JS, Bootstrap etc...
- Users are provided an image upload option on the screen by clicking on that user can upload the images they want.
- After processing the input image. Screen will be loaded with all the different object images and their corresponding informative links.
- Users can tap on any link they want to explore.

**5.5 Performance Requirements:**

- Model should run very fast so that users have a good experience.
- We expected that our model could process an image in half a second.
- CNN requires very less time but google api response time depends upon its server availability. So this may vary some times.
- And input images should be clear to identify the images completely blurred images are difficult to identify.
- To achieve all above requirements the system should have a good GPU, at least 16GB RAM, 1TB Hard disk.

## 5.6 Design Constraints:

- Training and validating system should be fast with all good resources.
- Network should identify as many objects as possible from the given image.
- Input image should be clear minimum resolution of 150*150 pixels.
- Small objects in an object are difficult to identify and objects in image which are not in the dataset are not identified
- There should not be dependency between the server and client. A client should be able request server through IP or hostnames
- The user should be connected to internet to send the requests and get the response.
- The user should be able to upload an image or capture an image through the inbuilt camera.
- User should be able to crop the region of interest and send request to the server.

# CHAPTER 6

# SYSTEM REQUIREMENTS SPECIFICATION

## 6.1 Hardware Requirements:

This section contains the functional requirements of our project. Our code was tested on Google Colab

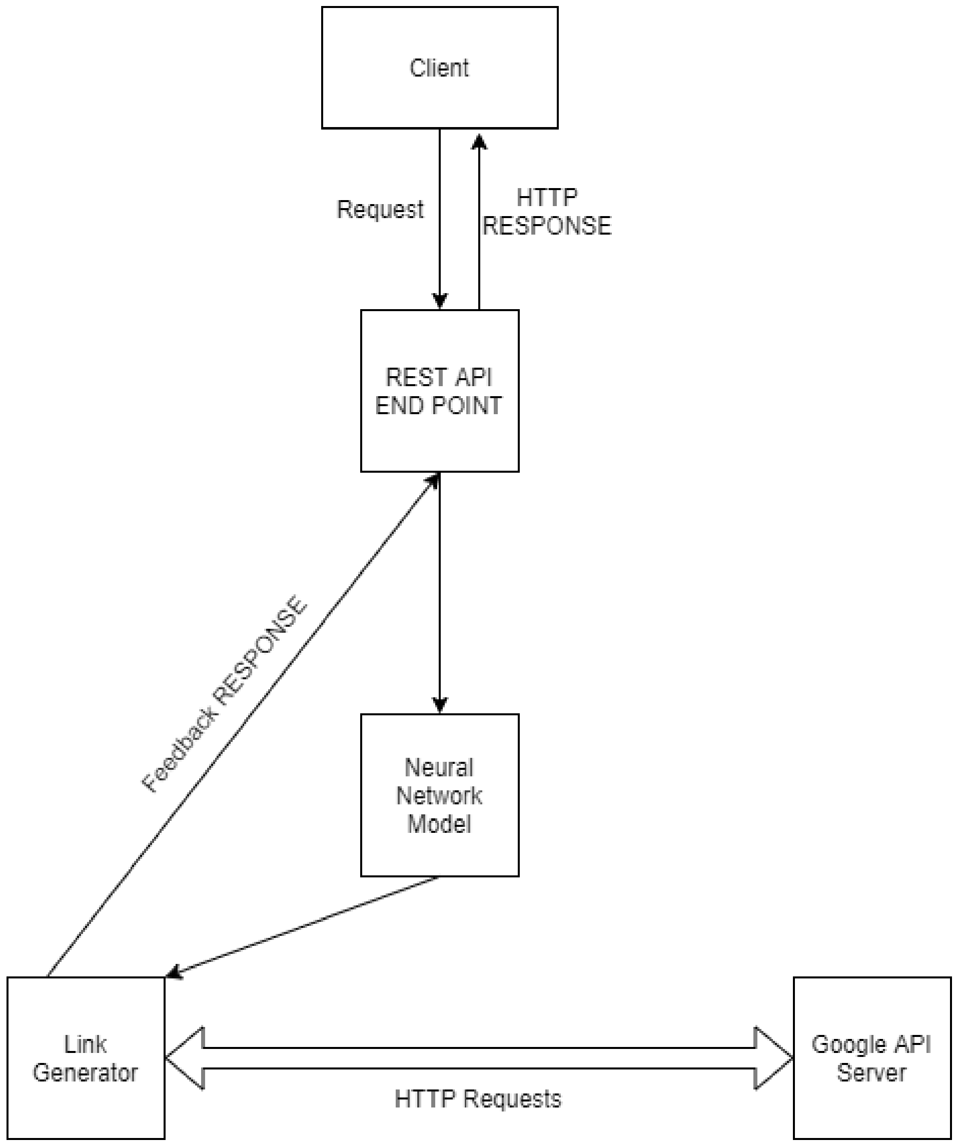
- Tesla CPU
- CUDA Supported NVIDIA GPU 4GB
- 16GB Ram
- 1TB Disk

## 6.2 Software Requirements:

- Jupyter Notebook
- Python-3.7+
- Numpy
- OpenCV
- Torch - 0.4.1
- TorchVision - 0.2.1
- Keras
- Tqdm
- Pillow
- Yacs
- Requests

# CHAPTER 7

## SYSTEM DESIGN



**Our system design has two parts.**

The first part contains a Client – Server Interface where a user can upload the image and server receives the image for processing. The server has the tasks of cross verifying the image size and format with the required specification. After pre-processing server send the image to SSD model.

The second part of the design has the convolutional neural network model SSD. SSD model after reading the image using OpenCV parses the image through the neural network layers. The model generates the bounding box coordinates and it is sent back to the server.

The server after receiving the image takes the bounding box and annotations. Each bounding box annotation part of the image will be cropped and searched in the google image API for specific information about the object detected. Client web framework receives the annotations and information about each object in the image. It will be parsed to generate links and bounding boxes using different colors using HTML and CSS.

# Chapter 8

# Low Level Design

Our project is divided into three parts

- Object Detection model Server
- Information Server
- Frontend HTML

## 8.1 Object Detection Model Deployed in the backend python server.

Single Shot Multi Box Detector: SSD Model is built using Pytorch ML Library. It is composed of two parts namely:

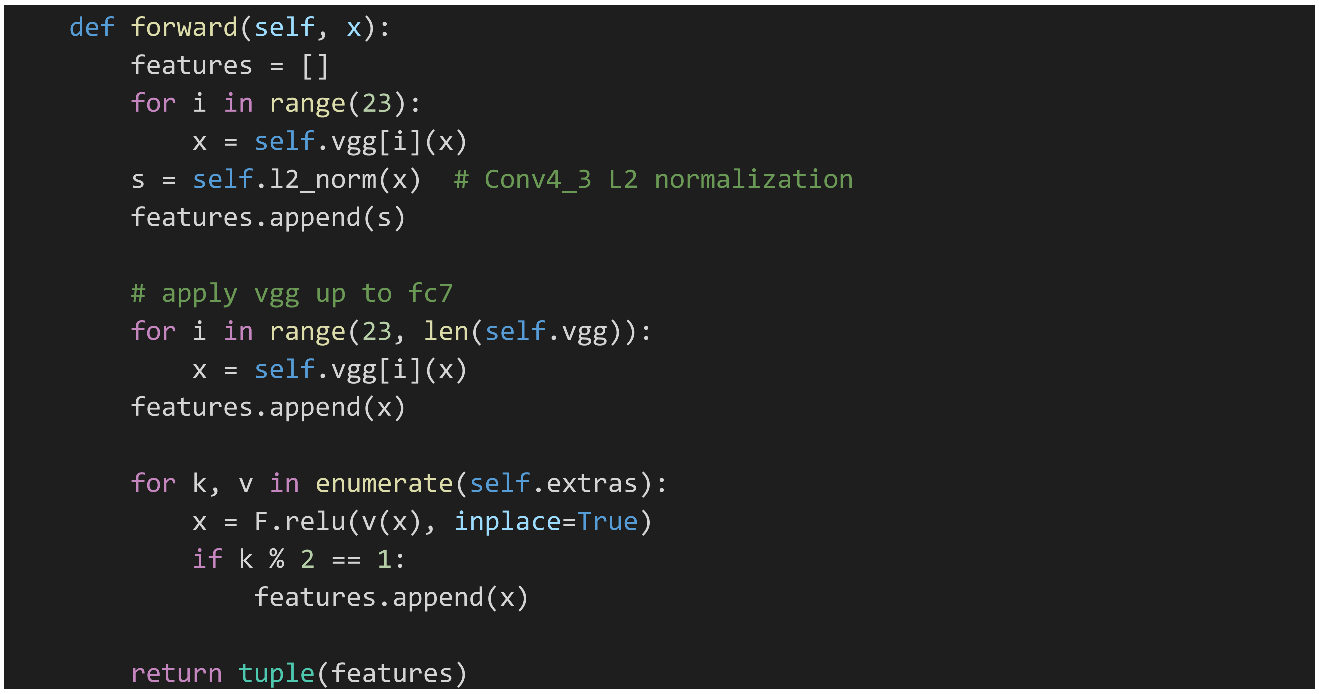
1. Backbone neural network which extracts feature maps.
    a. VGG 16 CNN is used as to extract the feature maps.

```python
class VGG(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        size = cfg.INPUT.IMAGE_SIZE
        vgg_config = vgg_base[str(size)]
        extras_config = extras_base[str(size)]

        self.vgg = nn.ModuleList(add_vgg(vgg_config))
        self.extras = nn.ModuleList(add_extras(extras_config, i=1024, size=size))
        self.l2_norm = L2Norm(512, scale=20)
        self.reset_parameters()

    def reset_parameters(self):
        for m in self.extras.modules():
            if isinstance(m, nn.Conv2d):
                nn.init.xavier_uniform_(m.weight)
                nn.init.zeros_(m.bias)

    def init_from_pretrain(self, state_dict):
        self.vgg.load_state_dict(state_dict)
```



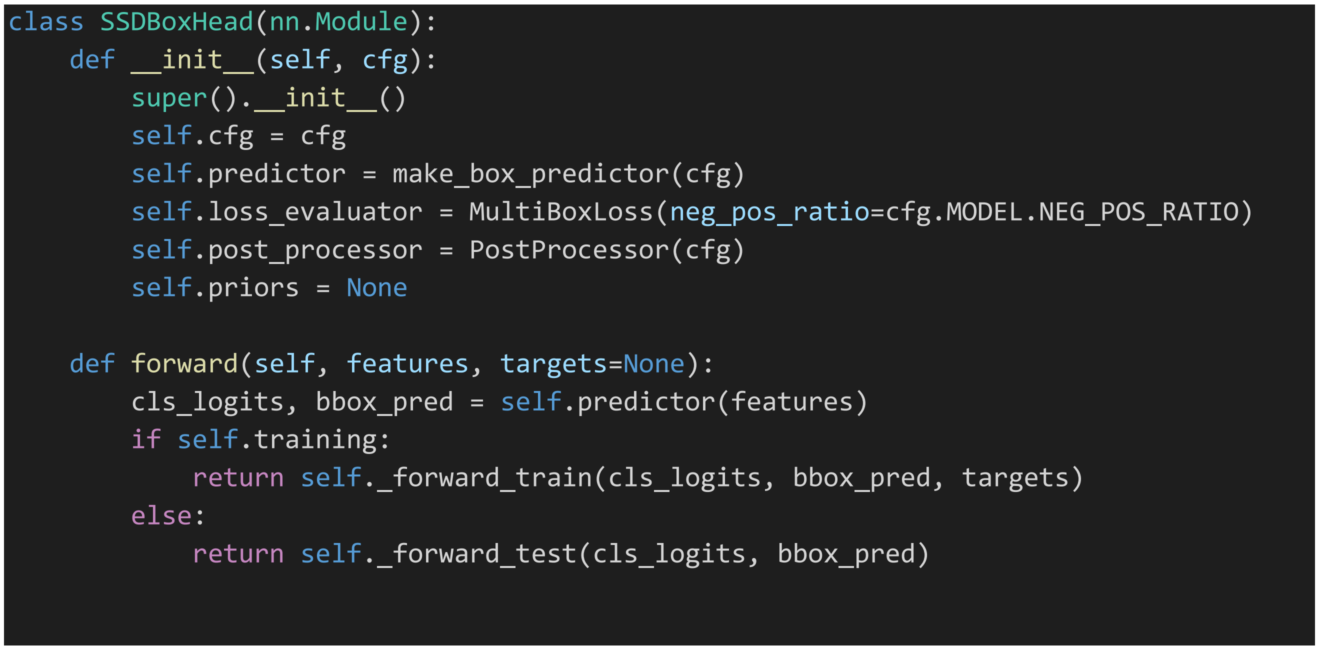
```
    def forward(self, x):
        features = []
        for i in range(23):
            x = self.vgg[i](x)
        s = self.l2_norm(x)  # Conv4_3 L2 normalization
        features.append(s)

        # apply vgg up to fc7
        for i in range(23, len(self.vgg)):
            x = self.vgg[i](x)
        features.append(x)

        for k, v in enumerate(self.extras):
            x = F.relu(v(x), inplace=True)
            if k % 2 == 1:
                features.append(x)

        return tuple(features)
```

2. Apply the convolutional filters to detect the objects



```
class SSDBoxHead(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.cfg = cfg
        self.predictor = make_box_predictor(cfg)
        self.loss_evaluator = MultiBoxLoss(neg_pos_ratio=cfg.MODEL.NEG_POS_RATIO)
        self.post_processor = PostProcessor(cfg)
        self.priors = None

    def forward(self, features, targets=None):
        cls_logits, bbox_pred = self.predictor(features)
        if self.training:
            return self._forward_train(cls_logits, bbox_pred, targets)
        else:
            return self._forward_test(cls_logits, bbox_pred)
```

The detections returned has each prediction which composed of bounding box and 80 scores for each class of COCO 2014 dataset. SSD uses Convolutional layer Conv4_3 which makes a total of 38*38*4 predictions.

## 8.2 Working of the Information Server:

It is a flask app which is deployed in pythonanywhere platform. Most importantly it contains google API key and the code to interface with the google vision project and code for post processing the response from vision model.

Google vision is a machine learning product developed by google. Which as many machine learning models in it like place search, OCR, google maps, Object detection, face detection etc.., All these models are deployed at google server to access these models or services we need to have API key which needs billing enabled for your project.

Among all these models we are using web detection model. This model is a sub part of google vision model. Which takes the image content as the input and search it among all the web pages and give top result for it. To do this we need to set an OS environment with GOOGLE_APPLICATION_CREDENTIALS" with the available key.

Then we take image name from the request sent by the user and pull that image from github where it was stored by object detection server. Then the image is read in 'rb' format using image libraries. Then we create a 'ImageAnnotatorClient' of vision model. On that we will call web detection model and send image content as input. Web detection will return top 10 results of different categories like best_guess_labels, pages_with_matching_images, full_matching_images, partial_matching_images and web_entities like Score, Description. In a json format like below:

```
 web_entities:
{
   entity_id: "/g/12v_d8cx1",
  score: 1.379182,
  description: "2008 HUMMER H2 SUT"
 },
```

partial_matching_images:

{

url: https://i.kinja-img.com/gawker-media/image/upload/cfill,fauto,flpr-ogressive,gcenter,h675,pg1,q80,w1200/18s4h3gx2rbvfjpg.jpg

},

best_guess_labels {

label: "hummer h2 2009",

language_code: "en"

}

Then take these responses and filter the information which is useful. After that we create a new json object with this information in a required format. Then we parse this response once again to take two links in each category and five links of pages matching images. Then take high score description, best label output and image class name and search in the Wikipedia for two sentences of description about the object. And add it to response then send the complete response to user interface.

Response from server:

{

"best_guess_labels": [ "hummer h2 2009" ],

"full_matching_images":["https://www.hdcarwallpapers.com/2009hummerh2sedona _metallicblackchrome-wallpapers.html"],

"pages_with_matching_images": [ "https://www.reviewcars.com/2009-hummer-h2"],

"partial_matching_images": ["https://i.kinja-img.com/gawker-media/image /upload/cfill,fauto,flprogressive,gcenter,h675,pg1,q80,w1200/18s4h3gx2rbvfjpg.jpg" ],

“web_entities": {
            "Description": [ "2008 HUMMER H2 SUT", "Hummer"],
      "Score": [ 1.3791823387145996,  1.0857000350952148 ]
},

  "wiki":
   [
   "The Hummer H2 is a large SUV that was marketed by Hummer and built in the AM General facility under contract from General Motors from 2002 to 2009. It is based on a modified GMT820 Chevrolet 2500 HD in front and 1500 frame in back."
  ]
}

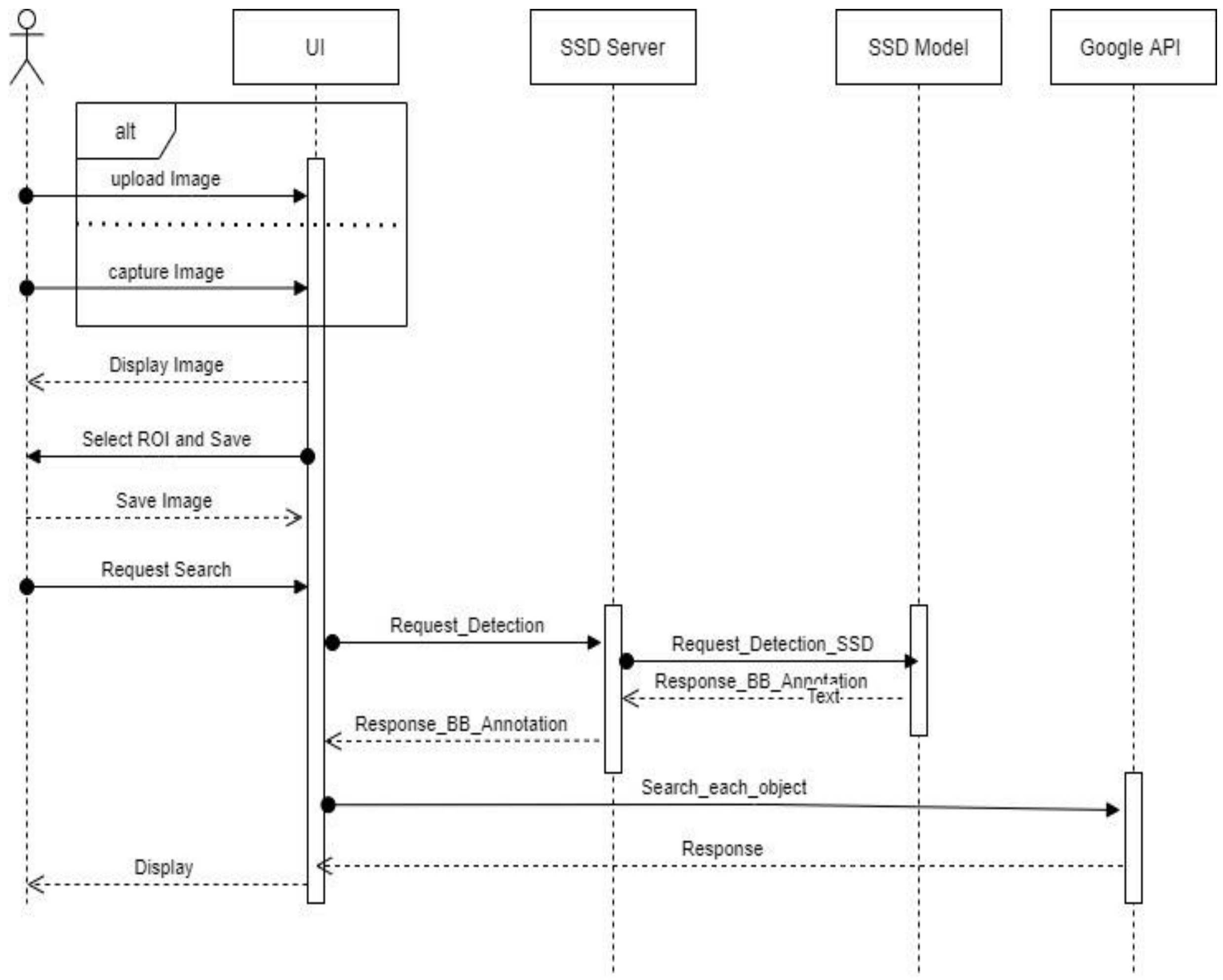
## 8.3 Request Handling by The Object Detection Server

- When we first run the server, we initialize with required dependencies and read the weights of SSD model and YOLO model and starts listening to the incoming requests.
- Once the request is received by the server, it reads all the bytes sent by the user and decodes it using base64 algorithm.
- After converting the server in its folder saves the image into a file
- A function called run_ssd() is called with the parameter as the file path of the image that was saved by the main function
- The function run_ssd() will read the image and passes the image through the SSD neural network which we trained and gets the detection. Based on the threshold we filter the results which are higher than the required threshold and append those to a json and send back the response to the main function in the dictionary format.
- The main function will now call the function run_yolo() with the parameter as the original image path.
- The run_yolo() function also passes the image into a pretrained neural network which was trained on COCO 2014 dataset.
- The results obtained from yolo model is filtered based on the threshold and are sent to the main function in the dictionary format.
- The results of both the model are passed to a function called crop_images() .
- The crop_images will receive the detections of both the model. Based on the number of objects detected in each class we take the maximum detections from one of the models and append it to the output JSON. Example: If yolo detected 5 persons 1 bottle and SSD detected 4 persons, 1 bottle and 1 cat, the final combined result would be like 5 persons of yolo detection,1 bottle of SSD detection, 1 cat of SSD Detection.

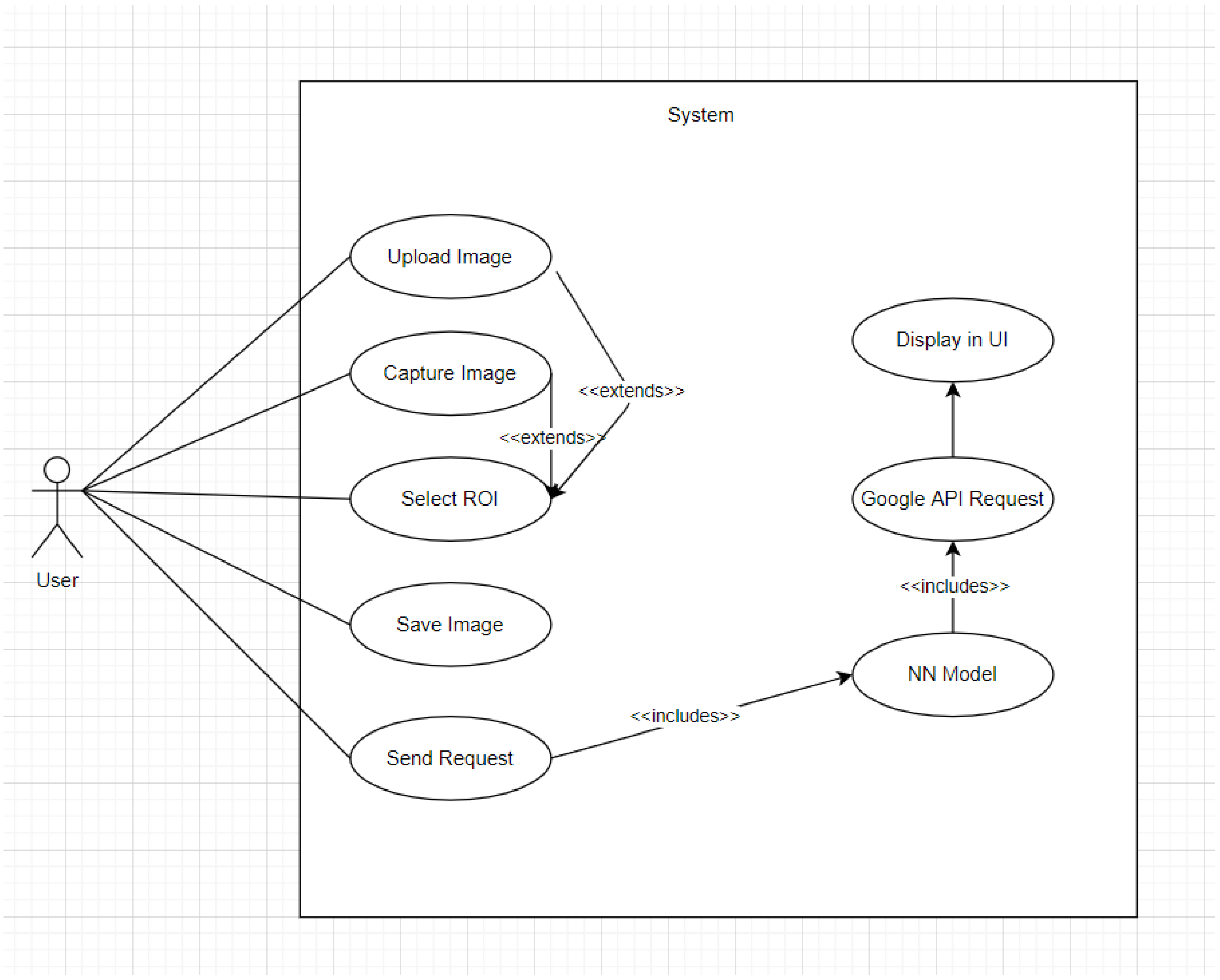
- After appending all the detections to output JSON, we parse it to crop the original image into the images and save the file with the class name followed by timestamp.

- Once all the images have been saved, we call the push_github () function.
- The push_github () function will add all the newly cropped images and commits to the git repository which we created to push the images.
- The output json is then sent to the user to render the output in the HTML.
- The JavaScript ajax response received by the web browser handles all the creation of elements for the detected objects and show it to the user in a tabular format with the images.
- To improve the speed of the server we implemented the in-memory weights to handle requests much faster
- We have multiple threads running in parallel for receiving the requests from the user in a round robin fashion to serve the requests.
- In a GPU less environment with better CPU the model takes just the 1.1 seconds to detect all the images.
- The time constraint also comes into picture if the cropped images are also big in size so that when we do a git push it takes much more time to push all the images as well as the original image to the github to send back the response.
- Through the github we eliminate the dependency of the Object detection server, Information server and the user. So the user when uploads the image the cropped images will be uploaded to the github and the user references the images using the github.
- The conflict of each image in the class is also eliminated using the timestamp based filename to save the cropped images.

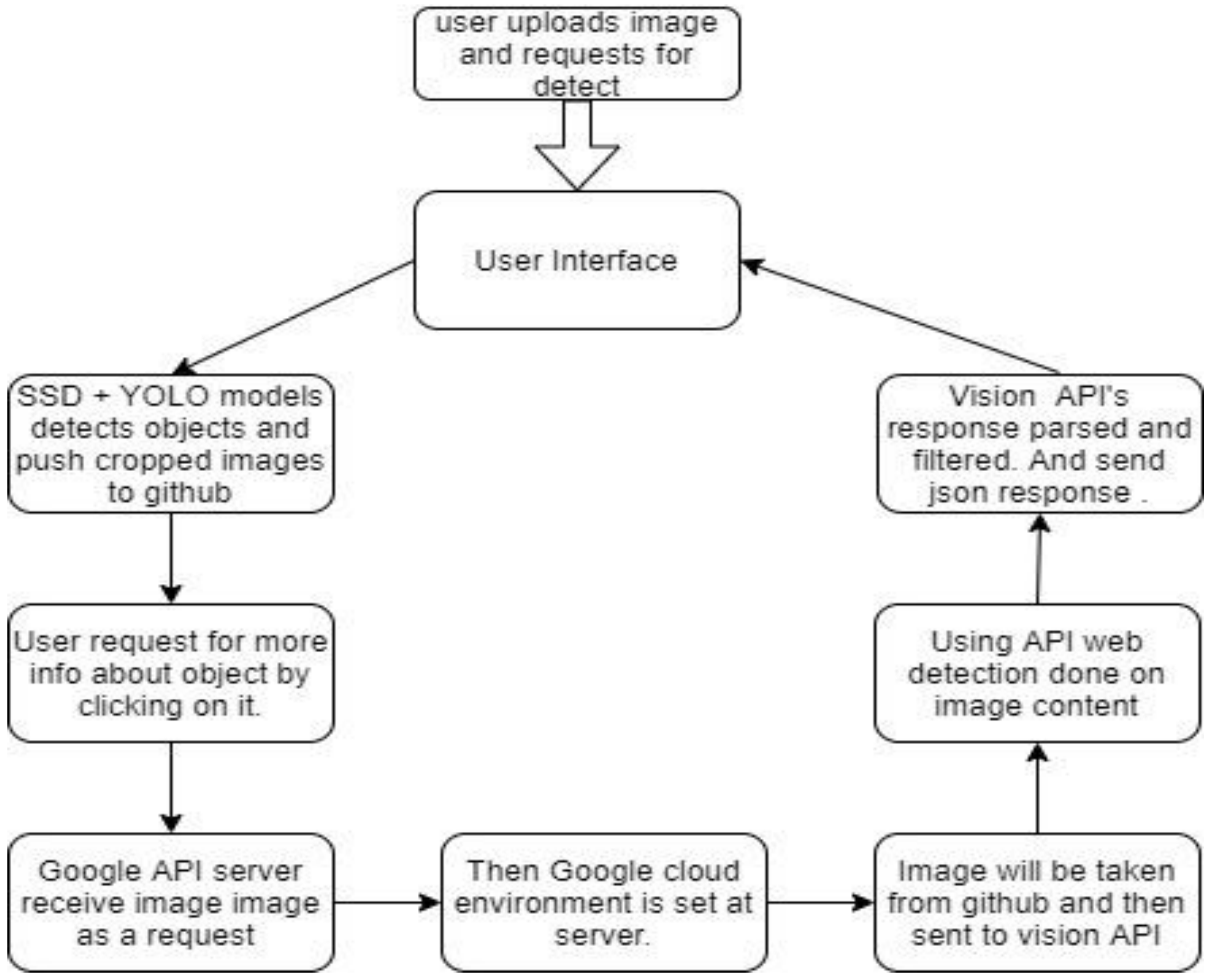
## 8.4 Sequence Diagram



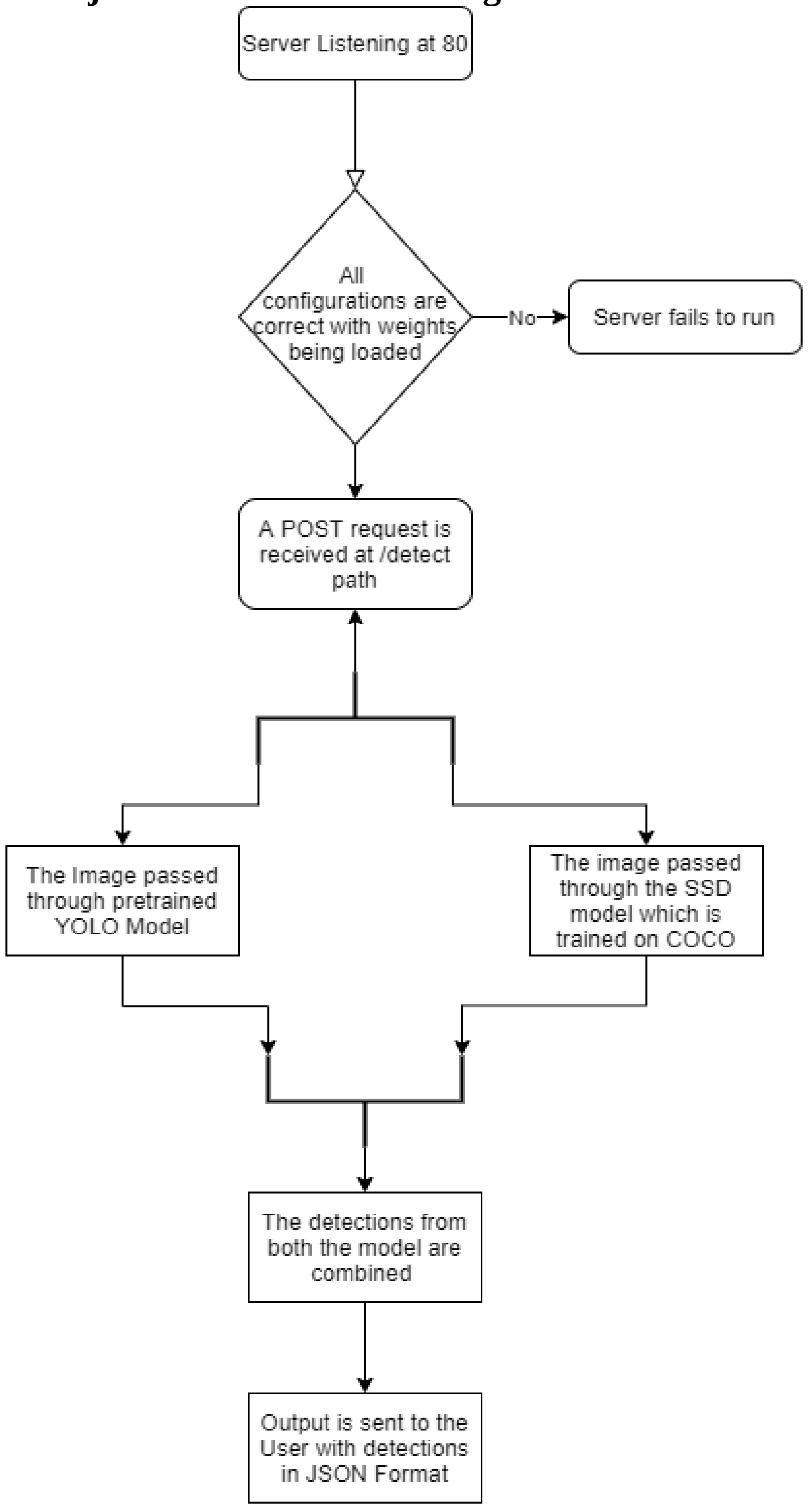
## 8.5 USE CASE DIAGRAM OF THE PROJECT



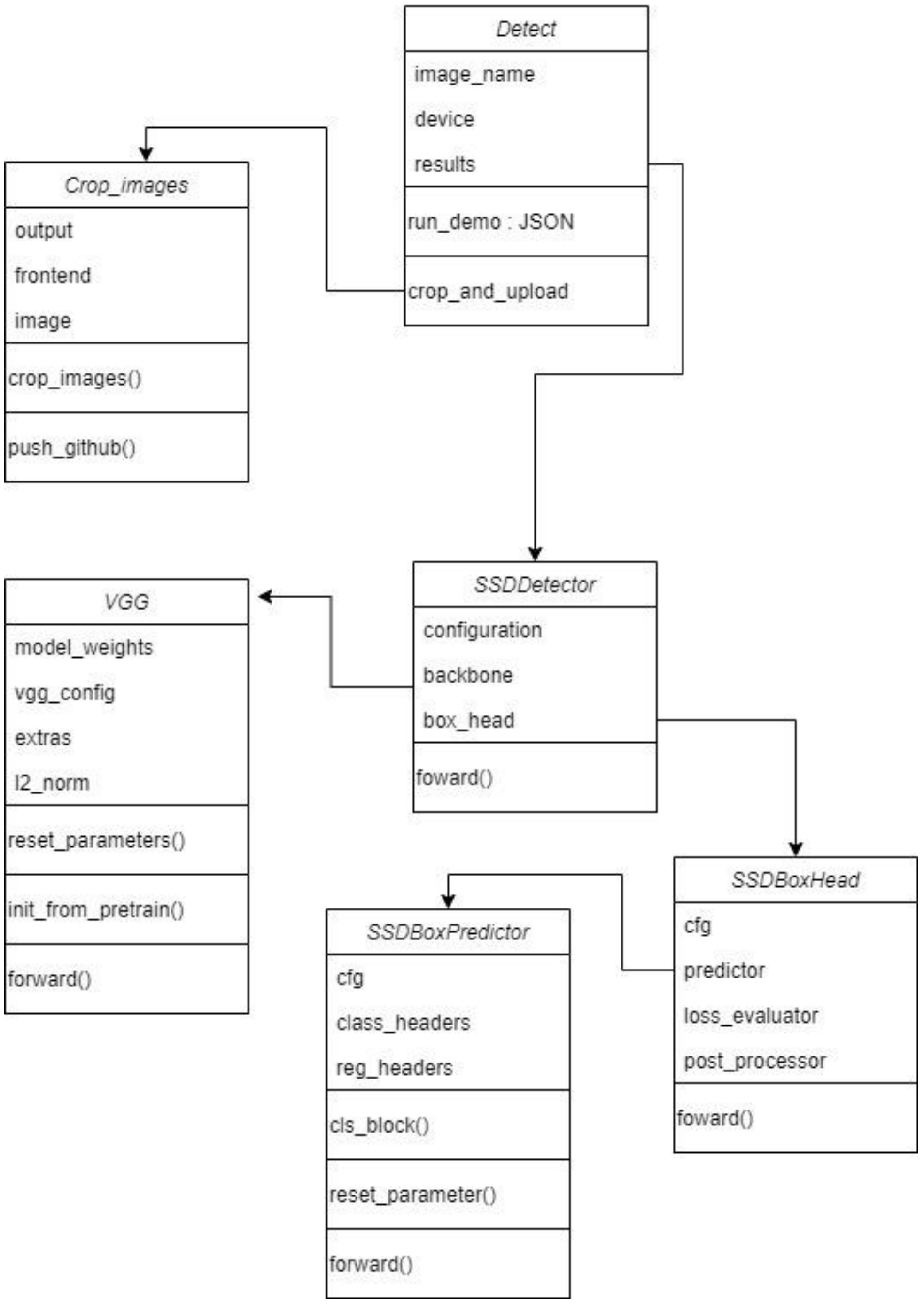
## 8.6 Information REST API server diagram


user uploads image and requests for detect
User Interface
SSD + YOLO models detects objects and push cropped images to github
User request for more info about object by clicking on it.
Google API server receive image image as a request
Then Google cloud environment is set at server.
Image will be taken from github and then sent to vision API
Using API web detection done on image content
Vision API's response parsed and filtered. And send json response .

## 8.7 Object Detection Server Diagram


Server Listening at 80
All configurations are correct with weights being loaded
No
Server fails to run
A POST request is received at /detect path
The Image passed through pretrained YOLO Model
The image passed through the SSD model which is trained on COCO
The detections from both the model are combined
Output is sent to the User with detections in JSON Format

## Class Diagram of SSD model



# Chapter 9

## USER GUIDE

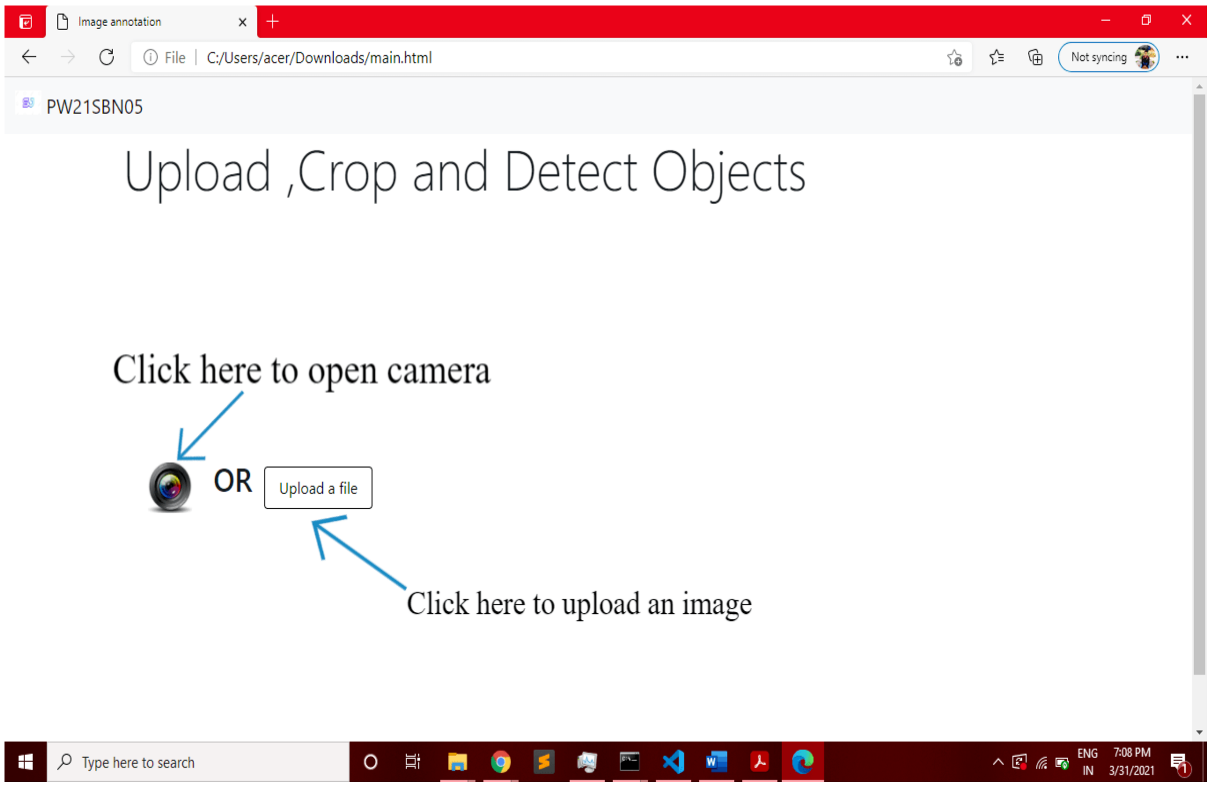


Image annotation
C:/Users/acer/Downloads/main.html
Not syncing
PW21SBN05
Upload ,Crop and Detect Objects
Click here to open camera
OR
Upload a file
Click here to upload an image
Type here to search

The user upon loading the webpage is shown this interface where there are two choices:

- Click photo using camera
- Upload an image

Click photo through the camera will prompts the user to allow the permission to access the web camera attached to the device. While in smartphones upload an image will also asks for the photo to be taken via the camera. The access to camera will generate the display of camera feed in a rectangular div upon which the button take snapshot will be visible through which we can take photo.

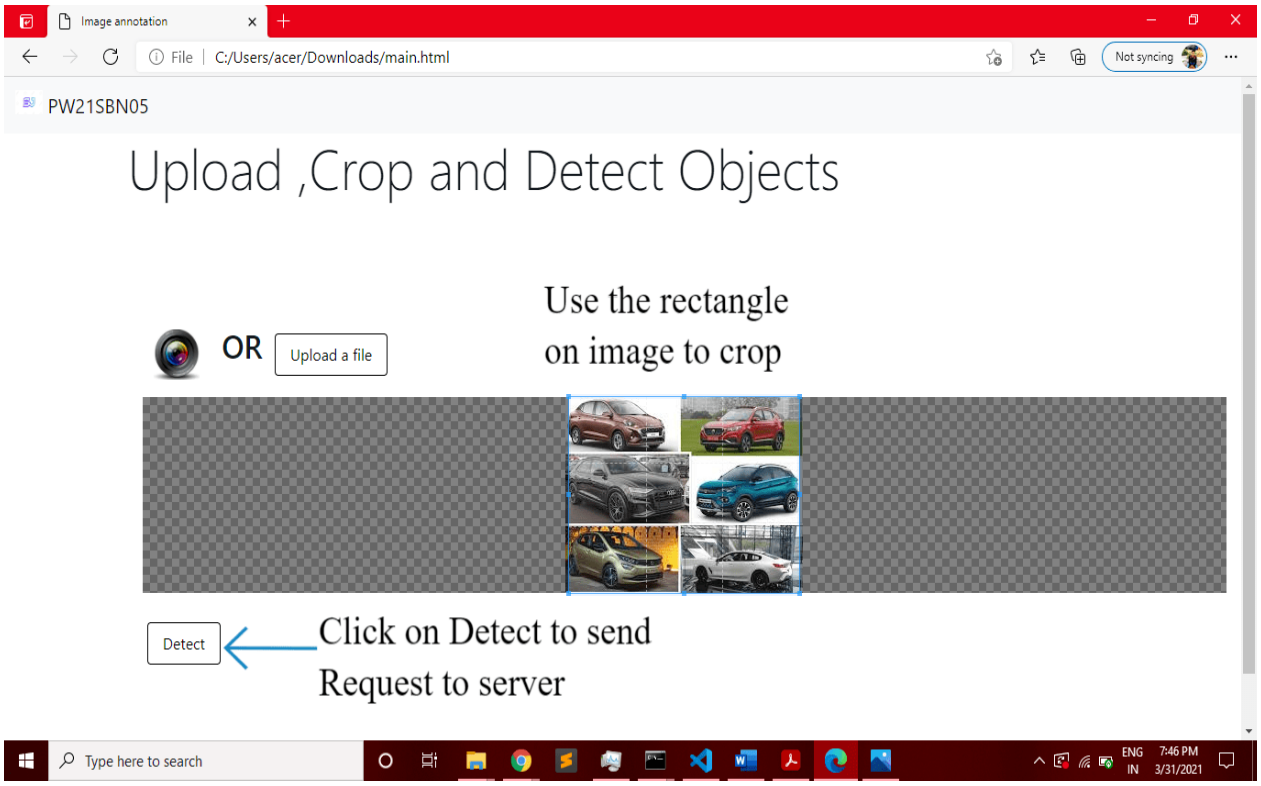


Image annotation
File | C:/Users/acer/Downloads/main.html
Not syncing
PW21SBN05
Upload ,Crop and Detect Objects
Use the rectangle on image to crop
OR
Upload a file
Detect
Click on Detect to send Request to server
Type here to search
ENG IN
7:46 PM 3/31/2021

After uploading an image or taking a photo through the camera user can crop the image based on his interest. The cropped image is saved internally in our canvas.

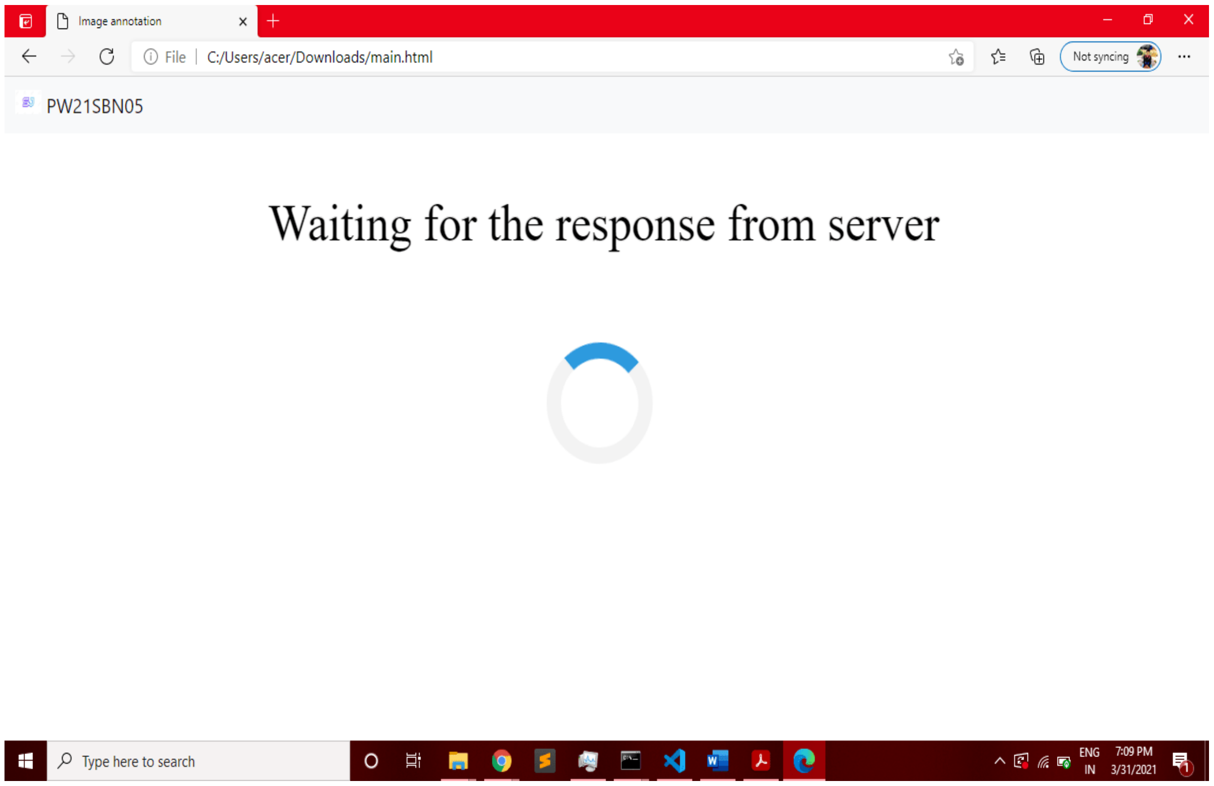
The click on the button called 'DETECT' will hide all the elements and pops up a loading animation which was created. In the background the listener on the button Detect will send the POST request to the user by reading all the bytes in the image cropped and sends to the user in a base64 encoded format through file upload request in the ajax.

Now, the user will receive all the object names in the JSON response as below:

```
{
        person123321.2335,
        person123351.2339,
        person12398.5594,
        dog12480.8894
}
```

Using all these objects the JavaScript attaches the links to the GITHUB to get these images and display in a tabular format to the user. User sees the loading animation disappearing and table with images is populated.

At this point of time, each Image will be having click listeners which sends request to the information server to the information from google, Wikipedia. Once the image is clicked the image will no longer have that listener and user cannot click again.



Waiting for the response from server

After the click on the Detect button the entire page is hidden and loading animation is displayed until we get a success or failure response. After the response obtained, we will tabulate all the results in the existing table and add click listeners based on the image name of each object received from the server.

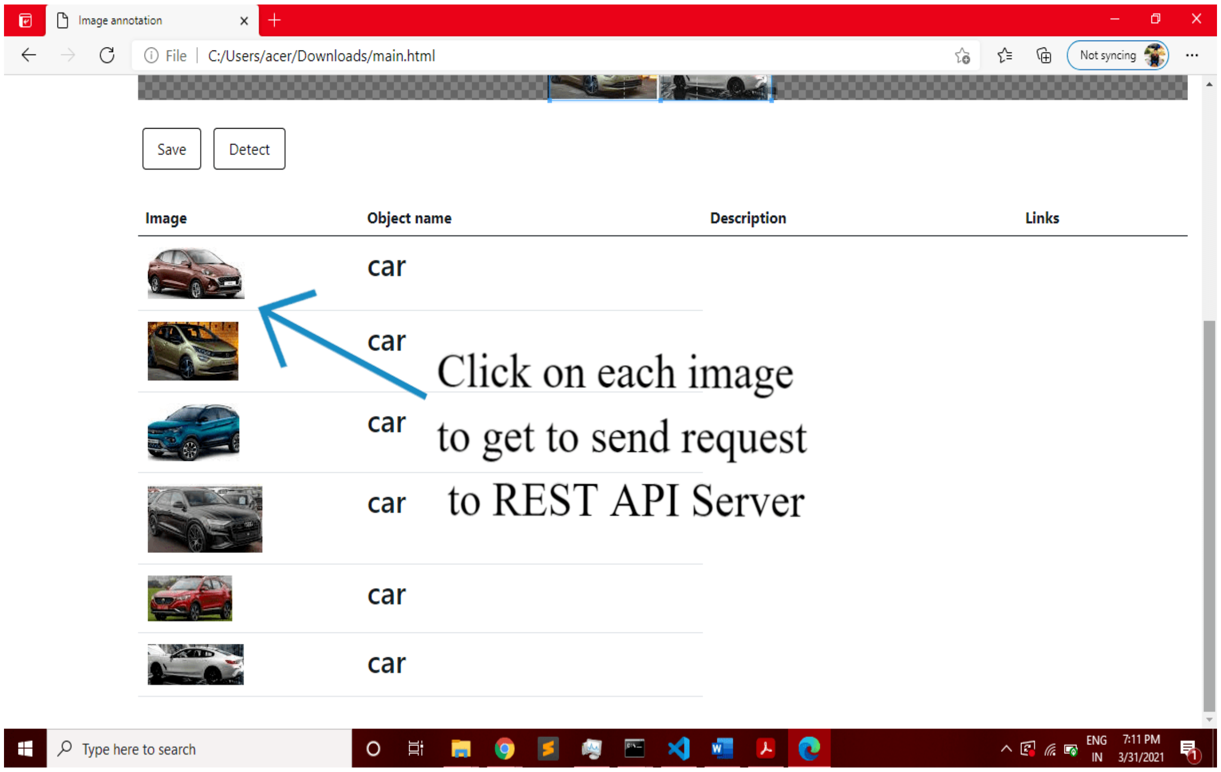
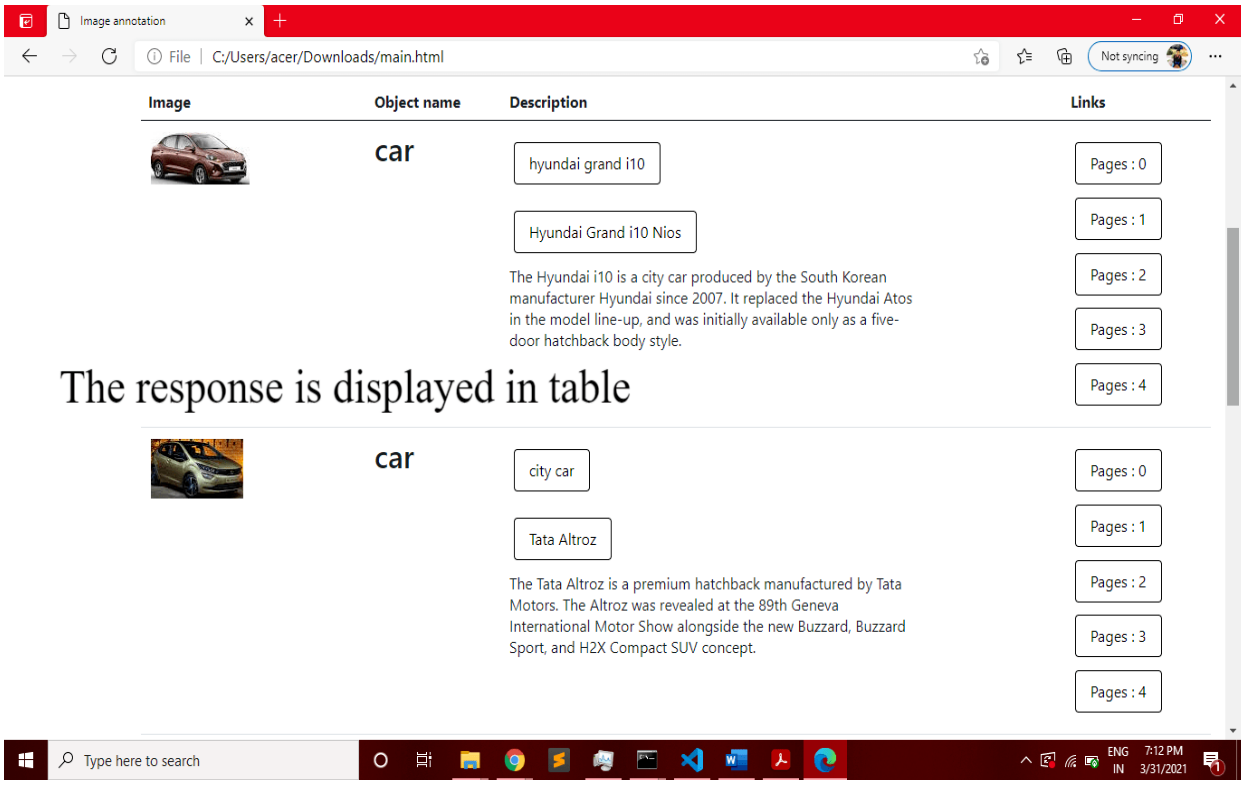


Image annotation
File | C:/Users/acer/Downloads/main.html
Not syncing
Save
Detect
Click on each image to get to send request to REST API Server
Type here to search
ENG IN
7:11 PM 3/31/2021

| Image | Object name | Description | Links |
|---|---|---|---|
| | car | | |
| | car | | |
| | car | | |
| | car | | |
| | car | | |
| | car | | |

Each object cropped is based on the size of the input image however we have put a limit on the maximum size of the image in the table to be 150 * 150 so that the consistency of the table is not disturbed. The objects on hover have black boundary which gives a sense to the user that the element is clickable. Once the element is clicked, the image boundary disappears even on hover and the image cannot be clicked after that. After the response is received, the results are appended to the exact same row where the user clicked the image.



| Image | Object name | Description | Links |
|---|---|---|---|
| | car | hyundai grand i10<br>Hyundai Grand i10 Nios<br>The Hyundai i10 is a city car produced by the South Korean manufacturer Hyundai since 2007. It replaced the Hyundai Atos in the model line-up, and was initially available only as a five-door hatchback body style. | Pages : 0<br>Pages : 1<br>Pages : 2<br>Pages : 3<br>Pages : 4 |
| | car | city car<br>Tata Altroz<br>The Tata Altroz is a premium hatchback manufactured by Tata Motors. The Altroz was revealed at the 89th Geneva International Motor Show alongside the new Buzzard, Buzzard Sport, and H2X Compact SUV concept. | Pages : 0<br>Pages : 1<br>Pages : 2<br>Pages : 3<br>Pages : 4 |

The response is displayed in table

# CHAPTER 10

## Evaluation of SSD model

After training the model on COCO train split, we evaluate the weights obtained against the test split of COCO 2014 which has 40608 images. After evaluating the model, we obtained the following results.

| **Metric** | **IoU** | **Max Detection** | **Area** | **Value** |
|---|---|---|---|---|
| Average Precision | 0.5:0.95 | 100 | All | 0.47 |
| Average Precision | 0.5 | 100 | All | 0.65 |
| Average Precision | 0.75 | 100 | All | 0.478 |
| Average Precision | 0.5:0.95 | 100 | Small | 0.26 |
| Average Precision | 0.5:0.95 | 100 | Medium | 0.504 |
| Average Precision | 0.5:0.95 | 100 | Large | 0.544 |
| Average Recall | 0.5:0.95 | 1 | All | 0.3 |
| Average Recall | 0.5:0.95 | 10 | All | 0.43 |
| Average Recall | 0.5:0.95 | 100 | All | 0.45 |
| Average Recall | 0.5:0.95 | 100 | Small | 0.15 |
| Average Recall | 0.5:0.95 | 100 | Medium | 0.527 |
| Average Recall | 0.5:0.95 | 100 | Large | 0.677 |

# CHAPTER 11

## CONCLUSION AND FUTURE WORK

In the first phase of the capstone project, we have trained the SSD model on PASCAL VOC dataset. Our SSD model was built on pytorch library of python. Starting off with the Literature survey, followed by dataset collection and model training. In the second phase of our capstone project, we have trained our model on PASCAL VOC 2012 Dataset which had 20 classes. In the second phase of the capstone, we had trained our model on 80 different classes of objects through COCO 2014 dataset which has 328,000 images.

We evaluated our model on the validation set of COCO 2014 and got the expected accuracy of our model. With AP50 of 0.65 the SSD model is the best-in-class object detection model that has been developed so far. It beats all the existing object detection models with its accuracy and speed.

Our goal was to create two python servers one for object detection and another for the information retrieval from google and Wikipedia. Two python flask servers are REST API based servers which handles our requests and sends the appropriate response. We achieved all the specifications and the accuracy that the model should satisfy in a given environment.

We have implemented front-end User-interface in the form of a HTML page with the required javascript code which has the functionality to communicate with the server and render the results.

We have completed the project as per the requirements and specifications provided. We have built the full stack application which can be deployed easily through simple steps and low-cost in-house hardware. In this project we have explored the ways we detect the objects in an image through different neural networks and developed a proper full stack application with cloud computing in picture. Image Annotation and object detection has significant scope in the field of astronomy, self-driving cars, security systems, biological systems and much more. As a sub project of Digital Image Processing, lot of technological developments are seen in this field.

This study can also be extended to detect more and more objects through higher datasets, more computer power and more default bounding boxes in SSD algorithm. Higher accurate results can be obtained through additional convolutional filters for each class and generate bounding boxes for the same. SSD is also a robust model which can be trained on custom dataset with bounding box detection.

## REFERENCE / BIBLIOGRAPHY

[1] Karen Simoyan, Andrew Zisserman "Very Deep Convolutional Neural Networks for Large Scale Image Recognition" 2015

[2] Shaoquin Ren, Kaiming He, Ross Girshick, Jian Sun "Faster R-CNN Towards Real time object Detection" 2015

[3] Joseph Redmon, Santosh Divvalay, Ross Girshick, Ali Farshadi "You Only Look Once: Unified, Real-Time Object Detection" 2016

[4] Wei Liu, Dragomir Angelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng Yang Fu, Alexander C Berg "SSD :Single Shot MultiBox Detector" 2016

[5] Yong Lui, Ruiping Wang "Structural Inference Net: Object Detection Using Scene Level Context and Instance-level Relationship" 2018

[6] Vincent Dumoulin, Francesco Visin "A guide to convolution arithmetic for deep learning" 2018

[7] Karanbir Chahal, Kuntal Dey "A Survery of Modern Object Detection Literature using Deep Learning

[8] Bell, S., Zitnick, C.L., Bala, K., Girshick, R.: Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In: CVPR. (2016)

[9] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollar "Microsoft COCO: Common Objects in Context"

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan "PyTorch : An Imperative Style, High Performance Deep Learning Library"

[11] Mark Everingham, Luc Van Gool, Christopher K.I. Williams, John Winn, Andrew Zisserman "The PASCAL Visual Object Classes (VOC) Challenge"

[12] Tiago Carneiro, Raul Victor Medeiros, Thiago Nepomuceno, Gui-Bin Bian, Victor Hugo, Pedro Pedrosa Rebouqas Filho "Performance Analysis of Google Collaboratory as a Tool for Accelerated Deep Learning Applications"

[13] 3.6A PRACTICAL STUDY ABOUT THE GOOGLE VISION API October 6th 2016. Daniel Pedro Ferreira Lopes, Antonio J. R

# APPENDIX A DEFINITIONS AND ABBREVIATIONS

1. Definition:

- Annotation: Additional information about the objects.
- Neural Network: Interconnected network of perceptron's.
- SSD: Deep CNN for image classification and object detection model.
- pytorch: open source library built on python used for computer vision applications.
- Bounding Box: Two-dimensional coordinate system for identification of objects in an image.
- Regression: Set of process for estimating the relationships between a dependent variable and one or more independent variables.
- Reginal Proposal: Set of regions in an image which has probability of presence of an object.

2. Abbreviations:

- VGG: Visual Geometric Group
- CNN: Convolutional Neural Network
- RCNN: Region based – CNN
- F-RCNN: Faster - RCNN
- SSD: Single Shot Multi-box Detector
- API: Application Programming Interface
- mAP: Mean Average Precision
- FPS: Frames Per Second
- IoU: Inter Section of Union