

# OBJECT RECOGNITION AND CLASSIFICATION

A REPORT

Submitted By:

Rangoli Jaiswal – 191B199

Rishi Neelkanth – 191B201

Sejal Jain – 191B223

**Under the Guidance of:** Dr. Neelesh Kumar Jain



**AUGUST 2021 – DECEMBER 2021**

*Submitted in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

*in*

**COMPUTER SCIENCE & ENGINEERING**

Department of Computer Science & Engineering

JAYPEE UNIVERSITY OF ENGINEERING & TECHNOLOGY,

A-B ROAD, RAGHOGARH, DT. GUNA - 473226, M.P

## **Declaration by the student**

I hereby declare that the work reported in the major project report entitled as **OBJECT RECOGNITION AND CLASSIFICATION**, in partial fulfilment for the award of degree of **Bachelor of Technology**, submitted at Jaypee University of Engineering and Technology, Guna, as per best of my knowledge and belief there is no infringement of intellectual property right and copyright. In case of any violation I will solely be responsible.

Rangoli Jaiswal 191B199

Rishi Neelkanth 191B201

Sejal Jain 191B223

Department of Computer Science and  
Engineering, Jaypee University of Engineering  
And Technology, Guna, M.P, India

Date: 05-12-2021

## **CERTIFICATE**

This is to certify that the work titled “**Object Recognition and Classification**” submitted by Rangoli Jaiswal, Rishi Neelkanth , Sejal Jain in partial fulfillment for the award of degree of B.Tech of Jaypee University of Engineering & Technology, Guna has been carried out under my supervision. As per best of my knowledge and belief there is no infringement of intellectual property right and copyright.

Also, this work has not been submitted partially to any other University or Institute for the award of this or any other degree or diploma. In case of any violation concern students will solely be responsible.

Signature of Supervisor

(Dr. Neelesh Kumar Jain)

Date:

## **ACKNOWLEDGEMENT**

The Major Project was a different experience this time, We have invested a lot of time and efforts in completion of this project but this project would not have been possible without the kind support of various individuals and organizations, we hereby extend our sincerest gratitude to all of them. We are highly indebted to our faculty Dr. Neelesh Kumar Jain . We would like to thank him for his constant guidance and supervision. We would like to thank the JUET for providing us with such supportive and innovative environment facilities and lending us their support whenever required. We will also like to thank the whole as an organization for providing us with such opportunities. At last we all will like to thank and congratulate our fellow project members for their hard-work and dedication.

Signature of Students:

Rangoli Jaiswal

Rishi Neelkanth

Sejal Jain

Date:

## **EXECUTIVE SUMMARY**

The project is on real time object detection and identification of objects. It will be implemented in Python using Open CV library and pre trained Tensor flow frozen model. The algorithm used is SSD .

SSD also known as Single Shot Detector is a single convolutional neural network. It works in corporation of the extracted features and bounding boxes. SSD allows more aspect ratios for generating default bounding boxes around the objects detected. SSD boxes can wrap around the objects in a tighter and more accurate manner.

So, for this project we are using pre trained deep learning architecture based on tensor flow.

To make the project we need to carry out J processes image identification, image detection and image segmentation.

In image identification we simply classify or identify what object is in the image. For which we are using Mobile net deep learning algorithm and Image net dataset.

In object detection we simply detect the area where object is present in the image. For detection we are using SSD, Mobile net VJ and COCO Dataset.

In image segmentation it is the segmentation between foreground and background.

For implementation used Anaconda Package and Jupiter Notebook as ide and we require Open CV library which is installed through pip command and mat plot lib. pyplot library for static, animated and interactive visualizations in python.

Then using SSD mobile net VJ and tensor flow Frozen model, code the project further and in the end we will be able to detect and classify static picture, mp4 video as well as through webcam.

With the help of this project we will present the analysis and implementation of Real-Time Object detection using SSD which is one of the fastest object detection algorithms. The goal of this project is to analyze different models and their knowledge in this domain.

## **TABLE OF CONTENTS**

Declaration by Student	02
Certificate	03
Acknowledgement	04
Executive Summary	05
List of Figures	07
Chapter 1: Introduction	08
Chapter 2: Literature Survey	09
Chapter 3: Implementation and Training	11
Chapter 4: System Requirements	13
Chapter 5: Proposed Work	15
Chapter 6: Result and Analysis	17
Chapter 7: Conclusion and Future Work	22
Chapter 8: References	30
Student's Profile	31

## **LIST OF FIGURES**

FIGURE	TITLE	PAGE NO
Figure-01	CNN layer	13
Figure-02	CNN architecture	13
Figure-03	SSD architecture	16
Figure-04	Block Diagram	20
Figure-05	Flowchart	21
Figure-06	Anaconda navigator	22
Figure-07	Jupyter Notebook	23
Figure-08	Import of OpenCV	23
Figure-09	Code snippet	24
Figure-10	Input Image	24
Figure-11	Check Background colour	25
Figure-12	Output	25
Figure-13	Result of implementation	27
Figure-14	Result of implementation	27
Figure-15	Result of implementation	28
Figure-16	Result of implementation	28

## **CHAPTER 1: INTRODUCTION**

In current world scenario amount of image data in the world is growing exponentially. The majority of these photographs are, however, kept in online clouds. We require some knowledge of the data's contents in order to manage such massive amounts of data properly. Many activities that call for picture captioning or image identification benefit from automated image processing. Images files can contain objects that can be automatically located and recognised.

As with conventional object detectors, contemporary object detectors use feature extractor and feature classifier. In contrast to how they are perceived to be in modern object detectors, these two components can be comprehended separately.

These systems not only identify and categorise every object in an image, but they also localise each object individually by drawing a bounding box around it. Real-Time object detection is therefore more difficult to perform than image recognition using conventional computer vision detectors. The fastest of the other kinds is the SSD, commonly referred to as Single Shot Detectors, because it precisely detects the object. Although it is not the most accurate model compared to the others, its speed makes it useful for applications that demand quick detection, such as people counting, autonomous vehicles, and other similar tasks.

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 needs 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 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. An example of a bicycle detection in an image that specifies the locations of certain parts is shown in Figure 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.



## **CHAPTER 2: LITERATURE STUDY**

### **What is AI?**

Artificial intelligence generally refers to processes and algorithms that are able to simulate human intelligence, including mimicking cognitive functions such as perception, learning and problem solving. Machine learning and deep learning (DL) are subsets of AI.

Specific practical applications of AI include modern web search engines, personal assistant programs that understand spoken language, self-driving vehicles and recommendation engines, such as those used by Spotify and Netflix.

There are four levels or types of AI—two of which we have achieved, and two which remain theoretical at this stage

#### **4 types of AI**

In order from simplest to most advanced, the four types of AI include reactive machines, limited memory, theory of mind and self-awareness.

Reactive machines are able to perform basic operations based on some form of input. At this level of AI, no “learning” happens—the system is trained to do a particular task or set of tasks and never deviates from that. These are purely reactive machines that do not store inputs, have any ability to function outside of a particular context, or have the ability to evolve over time.

Examples of reactive machines include most recommendation engines, IBM’s Deep Blue chess AI, and Google’s AlphaGo AI (arguably the best Go player in the world).

Limited memory AI systems are able to store incoming data and data about any actions or decisions it makes, and then analyze that stored data in order to improve over time. This is where “machine learning” really begins, as limited memory is required in order for learning to happen.

Since limited memory AIs are able to improve over time, these are the most advanced AIs we have developed to date. Examples include self-driving vehicles, virtual voice assistants and chatbots.

Theory of mind is the first of the two more advanced and (currently) theoretical types of AI that we haven’t yet achieved. At this level, AIs would begin to understand human thoughts and emotions, and start to interact with us in a meaningful way. Here, the relationship between human and AI becomes reciprocal, rather than the simple one-way relationship humans have with various less advanced AIs now.

The “theory of mind” terminology comes from psychology, and in this case refers to an AI understanding that humans have thoughts and emotions which then, in turn, affect the AI’s behavior.

Self-awareness is considered the ultimate goal for many AI developers, wherein AIs have human-level consciousness, aware of themselves as beings in the world with similar desires and emotions as humans. As yet, self-aware AIs are purely the stuff of science fiction.

## What is ML?

**Machine learning** is a subset of AI that falls within the “limited memory” category in which the AI (machine) is able to learn and develop over time.

There are a variety of different machine learning algorithms, with the three primary types being supervised learning, unsupervised learning and reinforcement learning.

### 3 types of machine learning algorithms

As with the different types of AI, these different types of machine learning cover a range of complexity. And while there are several other types of machine learning algorithms, most are a combination of—or based on—these primary three.

**Supervised learning** is the simplest of these, and, like it says on the box, is when an AI is actively supervised throughout the learning process. Researchers or data scientists will provide the machine with a quantity of data to process and learn from, as well as some example results of what that data should produce (more formally referred to as inputs and desired outputs).

The result of supervised learning is an agent that can predict results based on new input data. The machine may continue to refine its learning by storing and continually re-analyzing these predictions, improving its accuracy over time.

Supervised machine learning applications include image-recognition, media recommendation systems, predictive analytics and spam detection.

**Unsupervised learning** involves no help from humans during the learning process. The agent is given a quantity of data to analyze, and independently identifies patterns in that data. This type of analysis can be extremely helpful, because machines can recognize more and different patterns in any given set of data than humans. Like supervised machine learning, unsupervised ML can learn and improve over time.

Unsupervised machine learning applications include things like determining customer segments in marketing data, medical imaging, and anomaly detection.

**Reinforcement learning** is the most complex of these three algorithms in that there is no data set provided to train the machine. Instead, the agent learns by interacting with the environment in which it is placed. It receives positive or negative rewards based on the actions it takes, and improves over time by refining its responses to maximize positive rewards.

Some applications of reinforcement learning include self-improving industrial robots, automated stock trading, advanced recommendation engines and bid optimization for maximizing ad spend.

## What is DL?

Deep learning (DL) is a subset of machine learning that attempts to emulate human neural networks, eliminating the need for pre-processed data. Deep learning algorithms are able to ingest, process and analyze vast quantities of unstructured data to learn without any human intervention.

As with other types of machine learning, a deep learning algorithm can improve over time.

Some practical applications of deep learning currently include developing computer vision, facial recognition and natural language processing.

Object Detection is applied in various fields to detect different types of objects accurately. There has been a lot of research in some of the very famous object detection algorithms like RCNN, ResNet, YOLO and R-FCN.

### 2.1 OpenCV

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it is integrated with various libraries, such as NumPy, Python is capable of processing the OpenCV array structure for analysis. To identify image pattern and its various features we use vector space and perform mathematical operations on these features.

The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence it's free for both **academic** and **commercial** use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. When OpenCV was designed the main focus was real-time applications for computational efficiency. All things are written in optimized C/C++ to take advantage of multi-core processing.

**Applications of OpenCV:** There are lots of applications which are solved using OpenCV, some of them are listed below

- face recognition
- Automated inspection and surveillance
- number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Interactive art installations
- Anomaly (defect) detection in the manufacturing process (the odd defective products)
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control
- object recognition
- Medical image analysis
- Movies – 3D structure from motion
- TV Channels advertisement recognition

## OpenCV Functionality

- Image/video I/O, processing, display (core, imgproc, highgui)
- Object/feature detection (objdetect, features2d, nonfree)
- Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
- Computational photography (photo, video, superres)
- Machine learning & clustering (ml, flann)
- CUDA acceleration (gpu)

## Image-Processing

Image processing is a method to perform some operations on an image, in order to get an enhanced image and or to extract some useful information from it. If we talk about the basic definition of image processing then “Image processing is the analysis and manipulation of a digitized image, especially in order to improve its quality”. Digital-Image.

An image may be defined as a two-dimensional function  $f(x, y)$ , where  $x$  and  $y$  are spatial(plane) coordinates, and the amplitude of fat any pair of coordinates  $(x, y)$  is called the intensity or grey level of the image at that point.

In another word An image is nothing more than a two-dimensional matrix (3-D in case of coloured images) which is defined by the mathematical function  $f(x, y)$  at any point is giving the pixel value at that point of an image, the pixel value describes how bright that pixel is, and what colour it should be.

Image processing is basically signal processing in which input is an image and output is image or characteristics according to requirement associated with that image.

Image processing basically includes the following three steps:

1. Importing the image
2. Analysing and manipulating the image
3. Output in which result can be altered image or report that is based on image analysis

## 2.1 CNN

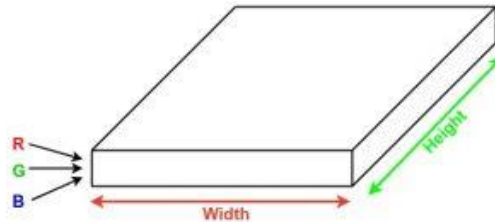
When it comes to Machine Learning, Artificial Neural Networks perform really well. Artificial Neural Networks are used in various classification tasks like image, audio, words. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolution Neural networks. In this blog, we are going to build a basic building block for CNN.

Before diving into the Convolution Neural Network, let us first revisit some concepts of Neural Network. In a regular Neural Network there are three types of layers:

1. **Input Layers:** It's the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in the case of an image).
2. **Hidden Layer:** The input from the Input layer is then feed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.

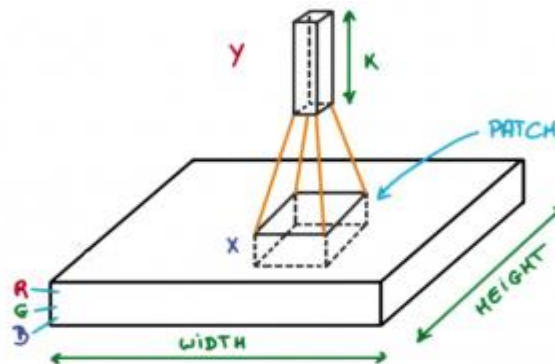
3. **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into the probability score of each class.

Convolution Neural Networks or convnets are neural networks that share their parameters. Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image), and height (as images generally have red, green, and blue channels).



**Figure -01 CNN LAYER**

Now imagine taking a small patch of this image and running a small neural network on it, with say,  $k$  outputs and represent them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different width, height, and depth. Instead of just R, G, and B channels now we have more channels but lesser width and height. This operation is called Convolution. If the patch size is the same as that of the image it will be a regular neural network. Because of this small patch, we have fewer weights.



**Figure – 02 CNN ARCHITECTURE**

Now let's talk about a bit of mathematics that is involved in the whole convolution process.

- Convolution layers consist of a set of learnable filters (a patch in the above image). Every filter has small width and height and the same depth as that of input volume (3 if the input layer is image input).
- For example, if we have to run convolution on an image with dimension  $34 \times 34 \times 3$ . The possible size of filters can be  $a \times a \times 3$ , where 'a' can be 3, 5, 7, etc but small as compared to image dimension.

- During forward pass, we slide each filter across the whole input volume step by step where each step is called stride (which can have value 2 or 3 or even 4 for high dimensional images) and compute the dot product between the weights of filters and patch from input volume.
- As we slide our filters we'll get a 2-D output for each filter and we'll stack them together and as a result, we'll get output volume having a depth equal to the number of filters. The network will learn all the filters.

## **2.2 RCNN**

Only 2000 regions from the image are extracted using a method called region proposals by Ross Girshick et al.

Rather than trying to classify a huge number of regions, the programme suggested using merely 2000 areas in order to classify a sizable number of regions. These 2000 region suggestions are square-shaped-bent/twisted and fed into a convolutional neural network, which outputs a 4096-dimensional feature vector. The CNN performs the function of a feature extractor, and the output dense layer is made up of the various features that were taken from the input image. The retrieved features are then passed into an SVM to categorise whether the object is present inside the suggested candidate region.

The issues with R-CNN are that it still takes a long time to train the network because it must categorise 2000 region proposals per image, and it cannot be used in real-world applications.

As a result, no learning is taking place at that point. The creation of poor region proposals could result from this.

## **2.3 YOLO (You Only Look Once)**

YOLO also known as You Only Look Once has the latest version as Yolo-v3 which is considered as custom CNN architecture, which is called DarkNet-53. The initial and first version, Yolo v1 architecture was inspired by Google Net which performed down sampling of the image detected and produced final predictions from a tensor. This tensor is obtained as in the Region of Interest pooling layer of the Faster R-CNN network. The next-generation Yolo v2 architecture used a 30-layer architecture, 19 layers from Darknet-19 and an additional 11 layers used significantly for the detection of objects. This new architecture provided a more accurate and faster object detection mechanism but it struggles with the detection of small objects in the region of interest detecting them incorrectly which can be a problem.

YOLO works by taking an image as input and splitting it into an  $S \times S$  grid, it takes  $m$  bounding boxes within each grid. For every bounding box, the network gives an output 'a' class probability and offset values for each bounding box formed. The bounding box having the class probability above a threshold value is selected and used to further locate the object within the image. YOLO is faster by order of magnitudes (45 frames per second) than other object detection algorithms present.

The limitation and disadvantage of YOLO algorithm is that it faces problem with the small objects within the image, for example, it might have difficulties in identifying a bird in the image. This is due to the spatial constraints of the YOLO algorithm.

## **2.4 ResNet**

ResNet helps to achieve excellent performance by training hundreds or thousands of layers. The performance of many Computer Vision applications other than image classification has been weakened such as face recognition and object detection by taking advantage of its powerful representational ability. Due to the vanishing gradient problem, which occurs when a gradient is repeatedly multiplied and becomes infinitesimally small, deep neural networks are challenging to train.

Therefore, performance starts rapidly declining or becoming saturated as the network gets deeper. ResNet's key concept is the introduction of a "identity shortcut link," which denotes the omission of one or more layers.

The size of these feature layers gradually decreased, enabling prediction of the detection on various scales. It has been demonstrated experimentally that when the input size is 300 or 320, the SSD's underlying convolution network is replaced by a residual network, which decreases rather than increases accuracy.

## **2.5 R-FCN**

R-FCN also has to obtain region proposals but in RFCN the fully convolutional layers after ROI pooling are removed.

After pooling, the FC layers' lengthy and time-consuming operation does not share the ROI, which slows RPN. The FC, layers multiply connections, which raises complexity. The average vote for each region proposal will be conducted using the same set of score maps, which is a straightforward computation. Convolutional feature maps that have been trained to recognise specific facets of each object make up these score maps. RFC outperforms Faster RCNN in terms of speed while using competitive mAP.

## **2.6 Fast RCNN and Faster RCNN**

Fast RCNN and Faster R have made further evolution in the field of object detection. To extract region-independent data, they employ convolutional layers that are initialised with pretraining for ImageNet classification.

Multi-layer perceptron (MLP) and features are used for classification.

Fast RCNN is faster at detecting objects than RCNN since it extracts features from the image before processing areas. Instead of developing a new model, it substitutes the SVM with a softmax layer, which aids in the extension of a neural network. The Region Proposal Network (RPN), which tries to learn the proposal of an object with the aid of feature maps, has taken the place of the Faster RCNN Selective Search approach. RPN receives the feature maps that were produced from CNN and uses them to suggest the regions. Such region proposals are to be generated for each location of the feature maps using K number of anchor boxes.

## **CHAPTER 3: IMPLEMENTATION AND TRAINING**

In this project, all the experiments were performed with the Tensorflow open-source deep learning framework, which is developed by the Google research team. We have used SSD know as Single Shot Multi-Box Detector with MobilNet v1 which is fastest model to detect objects.

By constructing m bounding boxes around each object in the image, the multi-box detector enables simultaneous detection of numerous things in the scene. Using a low-cost GPU and a pre-trained network with the COCO dataset, we developed this model. To further enhance the performance, we have increased our training set and tweaked a few parameters.

Convolutional layers, which are crucial for computer vision tasks but very expensive to compute in MobileNet V1, can be swapped out for depth-wise separable convolutions. The convolution layer's function is divided into two parts. The input is initially filtered by a depth-wise convolution layer.

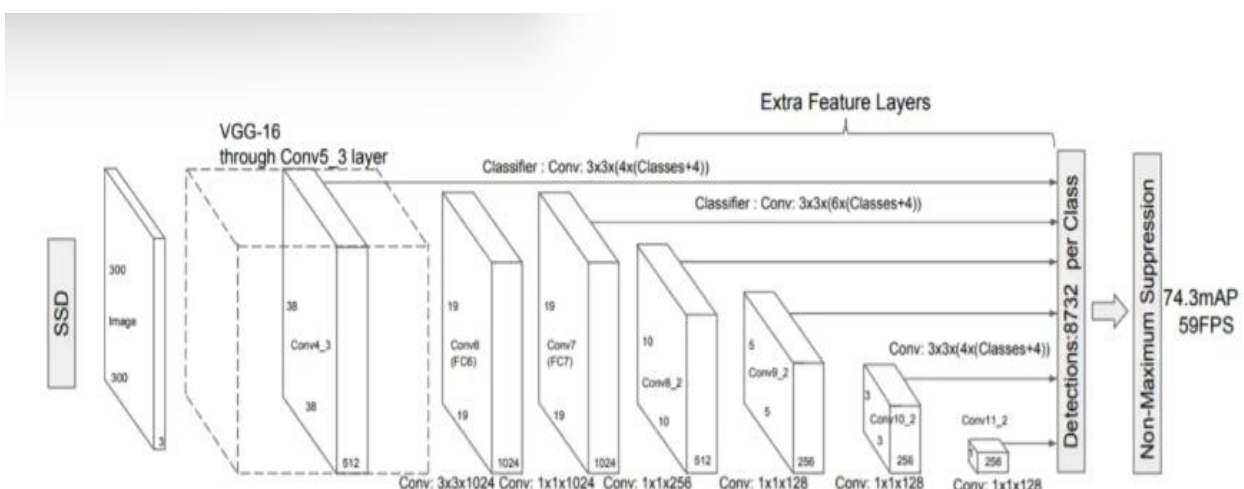
The new features are then created by combining these values in a 1X1 convolution. The depth wise separable convolution block is then formed by the depth wise and pointwise convolutions.

It does the same function as convolution method but moves considerably more quickly.

### **3.1 SSD**

A deep learning model called Single Shot MultiBox Detector is used to find items in images or videos. Single Shot Detector is a straightforward solution to the issue, but it has shown to be quite successful so far.

The Backbone Model and the SSD Head are the two parts of SSD. As a feature extractor, Backbone Model is a pre-trained image categorization network. The fully connected categorization layer is typically taken out of the model. The SSD Head is an additional set of convolutional layers that have been added to this backbone, and the outputs are interpreted as the bounding boxes and classes of objects in the spatial locations of the activations of the final layer.



**Figure - 03 SSD ARCHITECTURE**



The SSD object detection composes of 2 parts:

1. Extract feature maps, and
2. Apply convolution filters to detect objects.

SSD divides the image as grids, and each grid cell responsible for detecting objects in that region of the image. If there is no object detected then we output it as nothing or to be more precise we will put a "0" indicating that there is no object found.

SSD detects objects from a single layer. Actually, it uses multiple layers (multi-scale feature maps) for the detecting objects independently. As CNN reduces the spatial dimension gradually, the resolution of the feature maps also decrease. SSD uses lower resolution layers for the detect larger-scale objects. For example, the  $4 \times 4$  feature maps are used for the larger scale object.

Because of the cascade of pooling operations and non-linear activation, deep convolutional neural networks are able to categorise objects strongly against the realting transformation. The receptive field serves as the local search window in sliding window detection on SSD.

The stride of the convolution and the pooling process determine the limited resolution of SSD's search, much like all other sliding window techniques. SSD will get inaccurate data since the receptive field is not on target.

Deep convolutional neural networks have the ability to predict not only the class of an object but also its exact position. The bounding box can be represented by a vector of four floating-point values that SSD can map to. Since there are no longer any undesirable forms, the detection is significantly more accurate and requires far less processing power.

The classification task and the localisation task can share features because to SSD. The only difference between these two projects is in the very final layer. This helps the network to learn the characteristics while also substantially lowering the cost of computation.

## **ADVANTAGES OF SSD**

- Compared to two-shot detectors, SSD serves as their equivalent but has a lower overall cost. They accomplish substantially greater performance in situations where there are restricted resources available.
- It has a very slight reduction in exactness compared to other options. SSD300 recorded 59 FPS with mAP 74.3% while SSD500 recorded 22 FPS with mAP 76.9%.
- With lighter and shorter extractors, SSD can easily exceed Faster R-CNN and R-FCN in accuracy for big objects.
- To naturally manage objects of varying sizes, the Single Shot Detector network amalgamates predictions from numerous feature maps with varied resolutions.

## **CHAPTER 4: SYSTEM REQUIREMENTS**

Install Python on your computer system

1. Install Anaconda and anaconda prompt.
2. Python code implementation on Jupyter Notebook.

### **4.1 Steps to be followed:-**

- 1) Open anaconda navigator and prompt , set directory.
- 2) Open jupyter notebook for code implementation.
- 3) Install OpenCV , Matplotlib dependencies via pip.
- 4) Download the MobilenetV3.

#### **i.)Tensorflow:**

Tensor flow is an open-source software library for dataflow and differentiable programming across a range of tasks. It is an symbolic math library, and is also used for machine learning applications such as neural networks, etc.. It is used for both research and production by Google.

Tensorflow is developed by the Google Brain team for internal Google use. It is released under the Apache License 2.0 on November 9, 2015.

Tensorflow is Google Brain's second-generation system. 1st Version of tensorflow was released on February 11, 2017. While the reference implementation runs on single devices, Tensorflow can run on multiple CPU's and GPU (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on various platforms such as 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

The architecture of tensorflow allows the easy deployment of computation across a variety of platforms (CPU's, GPU's, TPU's), and from desktops - clusters of servers to mobile and edge devices.

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.

#### **pip install tensorflow -command**

## **ii. ) OpenCV**

OpenCV stands for Open Source Computer Vision Library, which is widely used for image recognition or identification. OpenCV is a library of programming functions mainly aimed on real time computer vision. originally developed by Intel, it is later supported by Willow Garage then Itseez. The library is a cross-platform and free to use under the open-source BSD license.

**pip install opencv-python –command**

## **iii.) Matplotlib:**

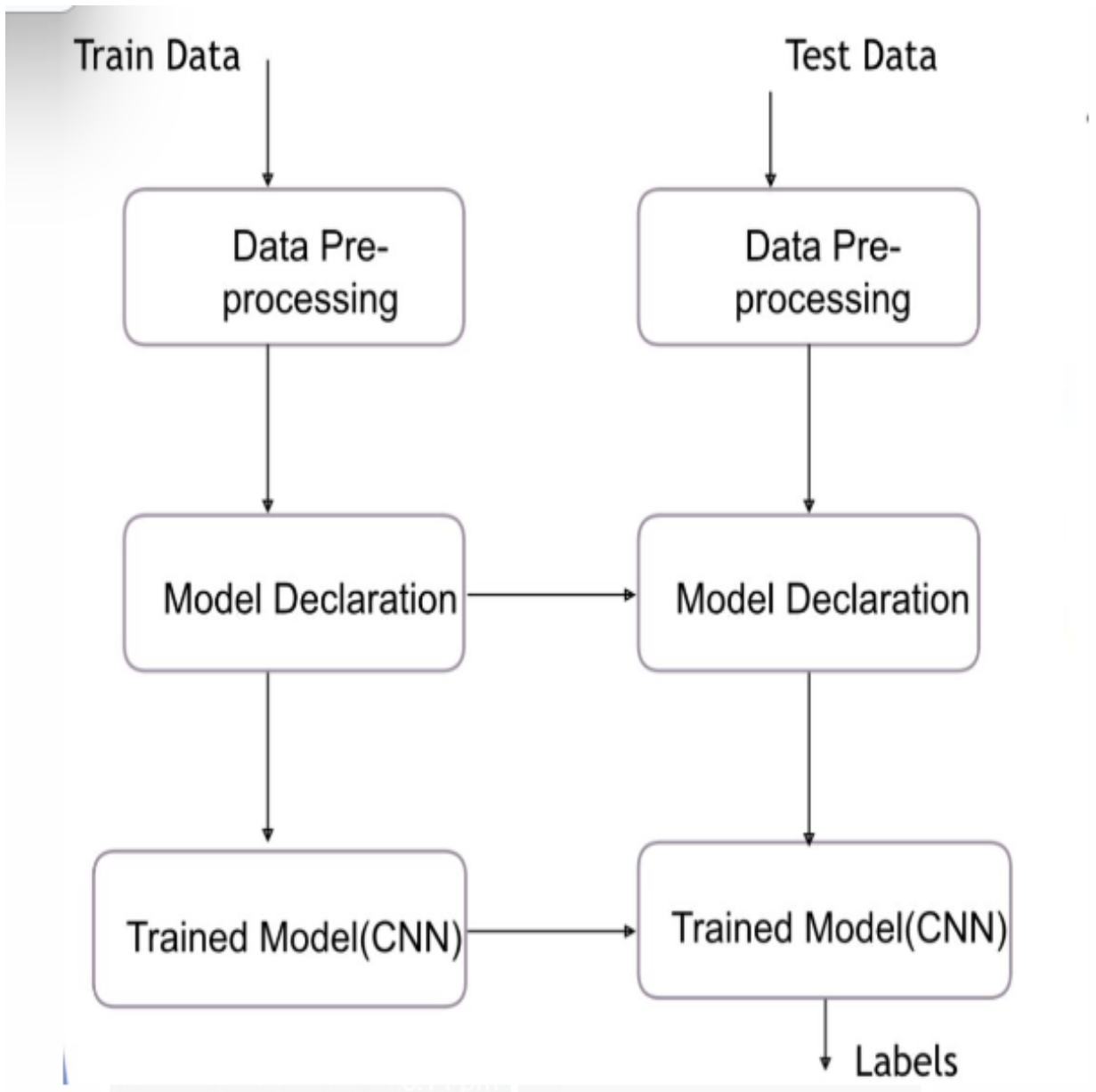
Matplotlib is a Python programming language plotting library that serves as a visualization utility. It provides an object-oriented API to use general-purpose GUI toolkits such as Tkinter, wxPython, Qt, or GTK+ to embed plots into applications.

Matplotlib is open source and we can use it freely. Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

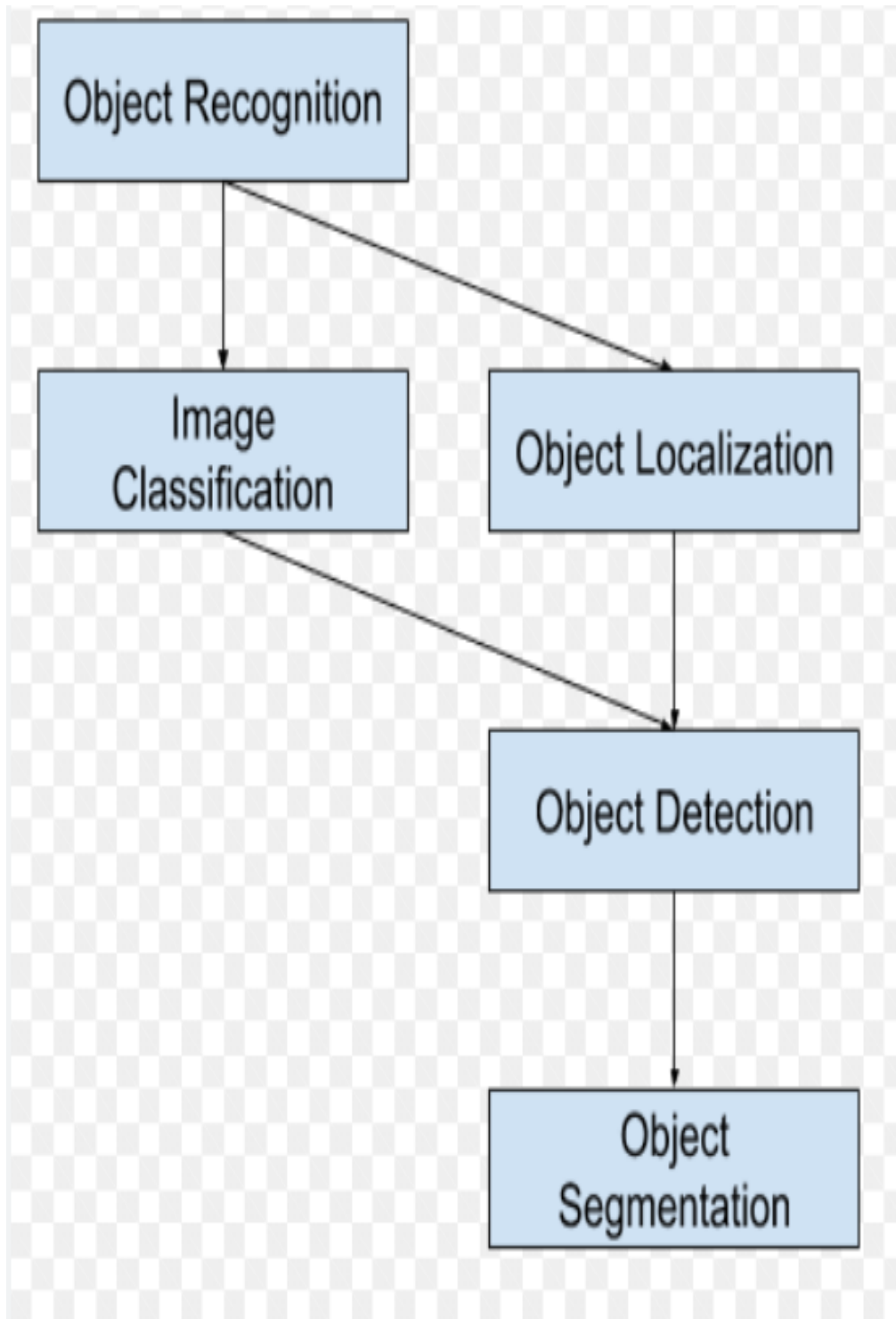
Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias:

**pip install matplotlib – command**

## **CHAPTER 5: PROPOSED WORK**

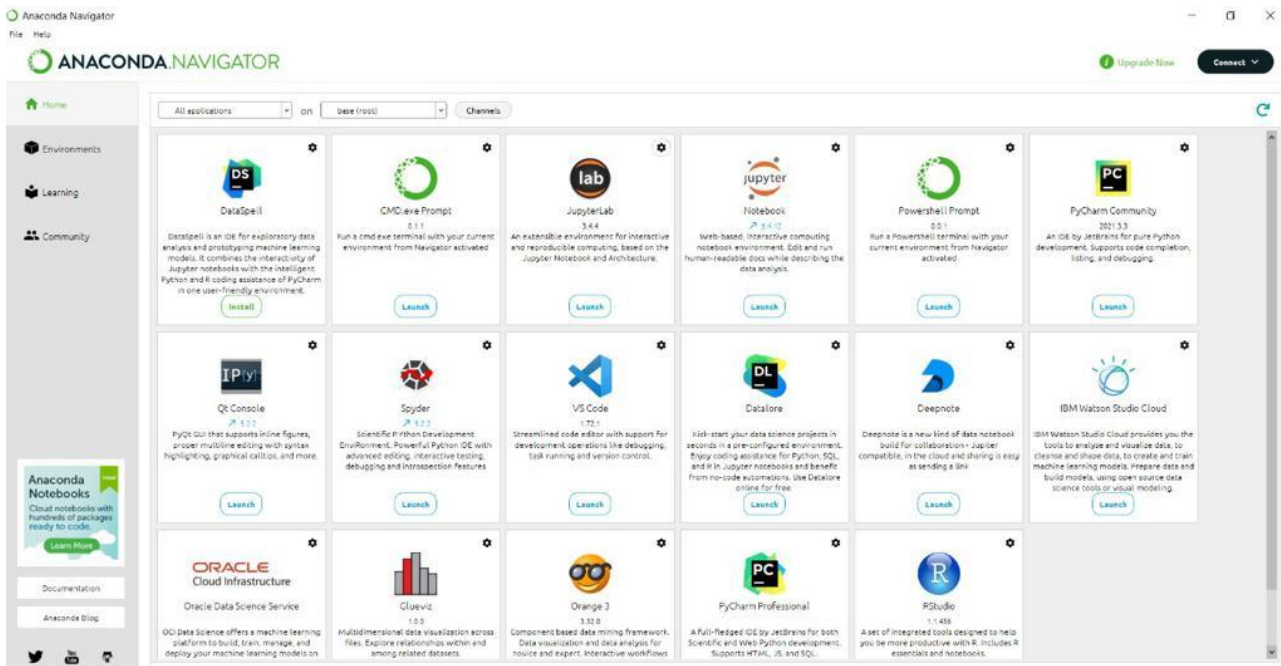


**Figure - 04 BLOCK DIAGRAM**



**Figure- 05 FLOW CHART**

## CHAPTER 6: RESULT AND ANALYSIS



**Figure -06 Anaconda navigator**

For implementation used Anaconda Package and Jupyter Notebook as ide and we require Open CV library which is installed through pip command and matplotlib, pyplot library for static, animated and interactive visualizations in python.

```
Anaconda Prompt (anaconda3) - jupyter notebook

(base) C:\Users\rishi>cd C:\MajorProject1

(base) C:\MajorProject1>jupyter notebook
[I 2022-12-04 19:38:46.463 LabApp] JupyterLab extension loaded from C:\Users\rishi\anaconda3\lib\site-packages\jupyterlab
[I 2022-12-04 19:38:46.464 LabApp] JupyterLab application directory is C:\Users\rishi\anaconda3\share\jupyter\lab
[I 19:38:46.469 NotebookApp] The port 8888 is already in use, trying another port.
[I 19:38:46.471 NotebookApp] Serving notebooks from local directory: C:\MajorProject1
[I 19:38:46.471 NotebookApp] Jupyter Notebook 6.4.12 is running at:
[I 19:38:46.471 NotebookApp] http://localhost:8889/?token=fed481ad2de1fedc035895d509b783d2de440429f850c3e3
[I 19:38:46.471 NotebookApp] or http://127.0.0.1:8889/?token=fed481ad2de1fedc035895d509b783d2de440429f850c3e3
[I 19:38:46.471 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 19:38:46.530 NotebookApp]

To access the notebook, open this file in a browser:
    file:///C:/Users/rishi/AppData/Roaming/jupyter/runtime/nbserver-17956-open.html
Or copy and paste one of these URLs:
    http://localhost:8889/?token=fed481ad2de1fedc035895d509b783d2de440429f850c3e3
    or http://127.0.0.1:8889/?token=fed481ad2de1fedc035895d509b783d2de440429f850c3e3
[I 19:39:00.280 NotebookApp] Kernel started: 74e03b71-4a97-4f9e-b3f7-656d8dee2e85, name: python3
[I 19:39:18.769 NotebookApp] Starting buffering for 74e03b71-4a97-4f9e-b3f7-656d8dee2e85:5364fb5a371c423b90e550e676f13fa
c
[I 19:39:19.130 NotebookApp] Kernel restarted: 74e03b71-4a97-4f9e-b3f7-656d8dee2e85
[I 19:39:19.145 NotebookApp] Restoring connection for 74e03b71-4a97-4f9e-b3f7-656d8dee2e85:5364fb5a371c423b90e550e676f13fa
c
[I 19:39:21.213 NotebookApp] Replaying 3 buffered messages
[I 19:41:00.230 NotebookApp] Saving file at /Major_Project1.ipynb
[I 19:41:16.115 NotebookApp] KernelRestarter: restarting kernel (1/5), keep random ports
kernel 74e03b71-4a97-4f9e-b3f7-656d8dee2e85 restarted
```

**Figure -07 Jupyter Notebook**

```
jupyter Major_Project1 Last Checkpoint: 16/11/2022 (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Dead kernel Trusted Python 3 (ipykernel)

In [1]: import cv2 # pip install opencv-python

In [2]: import matplotlib.pyplot as plt # pip install matplotlib

In [3]: config_file = 'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
        frozen_model = 'frozen_inference_graph.pb'

In [4]: model = cv2.dnn_DetectionModel(frozen_model,config_file)

In [5]: classLabels = [] ## empty list of python
        file_name = 'Labels.txt'
        with open(file_name,'rt') as fpt:
            classLabels = fpt.read().rstrip('\n').split('\n')

In [6]: print(classLabels)

['person', 'bicycle', 'car', 'motorbike', 'aeroplane', 'bus', 'train', 'truck', 'boat', 'traffic light', 'fire hydrant', 'stop
sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe', 'backpa
ck', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball
glove', 'skateboard', 'surfboard', 'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana',
apple', 'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'sofa', 'pottedplant', 'be
d', 'diningtable', 'toilet', 'tvmonitor', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone', 'microwave', 'oven', 'toaste
r', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear', 'hair drier', 'toothbrush']

In [7]: print(len(classLabels))

80

In [8]: model.setInputSize(320,320)
        model.setInputScale(1.0/127.5) ## 255/2 = 127.5
        model.setInputMean((127.5,127.5,127.5)) ## mobilenet => [-1,1]
        model.setInputSwapRB(True)

Out[8]: < cv2.dnn.Model 00000287179BA0D0>
```

**Figure -08 Import OpenCV**



Jupyter Major\_Project1 Last Checkpoint: 16/11/2022 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Dead kernel Trusted Python 3 (ipykernel)

VIDEO DEMO

```
In [16]: cap = cv2.VideoCapture('Sample_video.mp4')

# check if the video is opened corrently
if not cap.isOpened():
    cap=cv2.VideoCapture(0)
if not cap.isOpened():
    raise IOError("Correct open video")

font_scale = 3
font = cv2.FONT_HERSHEY_PLAIN

while True:
    ret, frame = cap.read()

    ClassIndex, confidece, bbox = model.detect(frame,confThreshold = 0.55)

    print(ClassIndex)
    if (len(ClassIndex)!=0):
        for ClassInd, conf, boxes in zip(ClassIndex.flatten(), confidece.flatten(), bbox):
            if(ClassInd<=80):
                cv2.rectangle(frame,boxes,(255,0,0), 2)
                cv2.putText(frame,classLabels[ClassInd-1],(boxes[0]+10,boxes[1]+40), font, fontScale = font_scale,color=(0,255,0)

    cv2.imshow('Object Detection Tutorial',frame)

    if cv2.waitKey(2) & 0xFF == ord('q'):
        break;

cap.release()
cv2.destroyAllWindows()
```

**Figure – 09 Project code Snippet**

Jupyter Major\_Project1 Last Checkpoint: 16/11/2022 (autosaved)

File Edit View Insert Cell Kernel Widgets Help


Dead kernel Trusted Python 3 (ipykernel)

read an image

```
In [9]: img = cv2.imread('man-car.jpg')
```

```
In [10]: plt.imshow(img) ## bgr
```

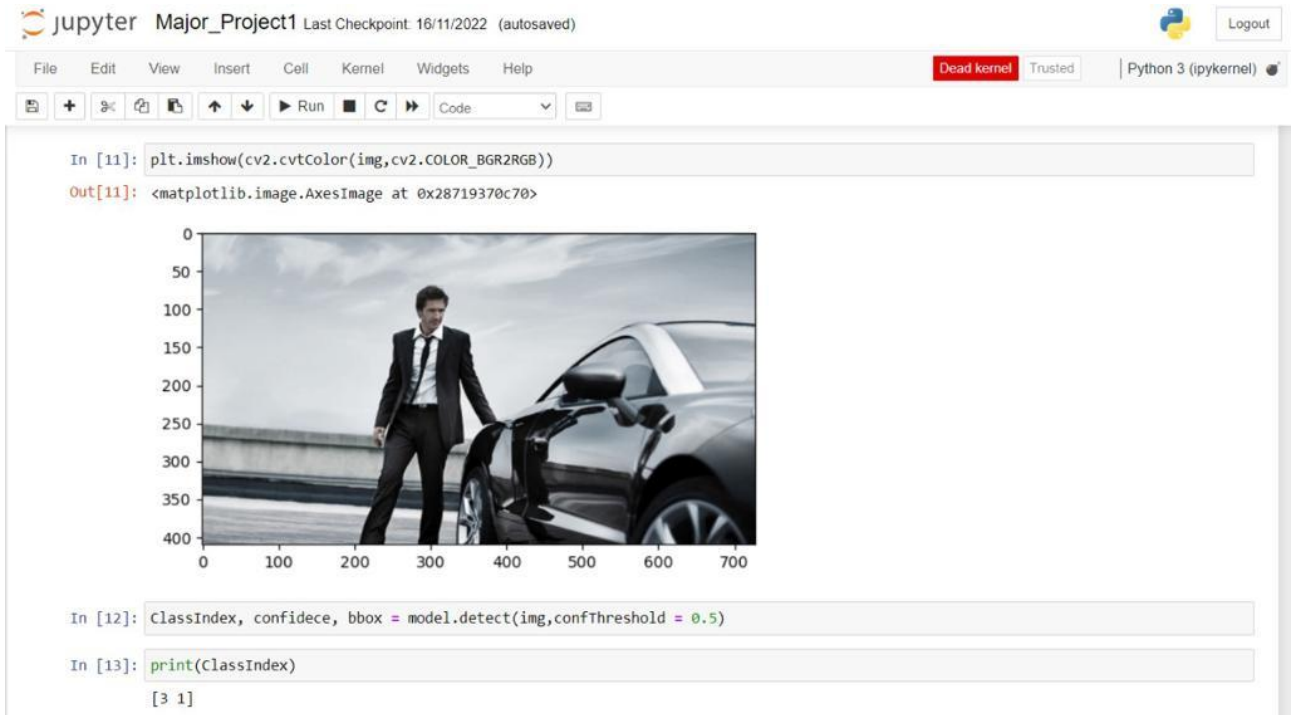
```
Out[10]: <matplotlib.image.AxesImage at 0x28718a08b80>
```



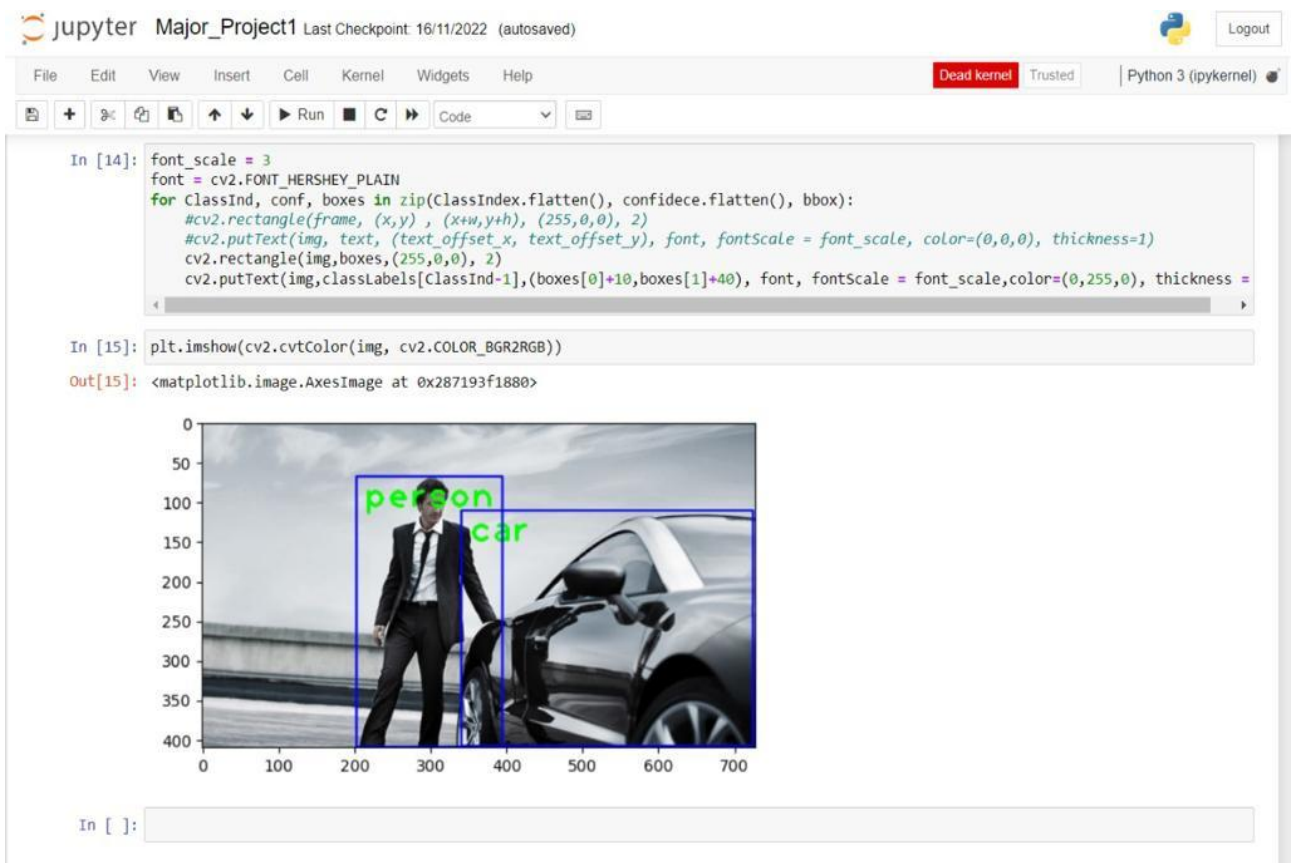
The image shows a man in a dark suit and tie standing next to a dark-colored car. The man is looking towards the camera. The car is a sedan, and its front end is visible. The background is a light, hazy sky. The image is displayed in a Jupyter notebook with axes, with the x-axis ranging from 0 to 700 and the y-axis ranging from 0 to 400.

**Figure – 10 Read Image**





**Figure – 11 Check Background Colour**



**Figure – 12 Output**

The suggested system has been tested numerous times with a variety of objects, and each time, it accurately detects and recognises the objects. The project was built on Python and assessed several times via the webcam. It has a respectable FPS. The input video is divided into the entire number of frames, and each frame is sent to our proprietary object detector for detection. After detection, the bounding box information is provided to the SSD algorithm, which then performs object detection. The live video was examined using our suggested system, which produced the output as bounding boxes with the class name and confidence score.

As evident from the above graphic, when compared to other object detection models, the SSD model with MobilNet v1 has the quickest GPU response time in milliseconds. When measured in GPU time, RCNN performed the slowest. This was done with a 300 dpi image resolution. This benefit makes SSD beneficial in applications like counting persons or items because it is the quickest model to recognise objects by taking the shortest amount of time. Additionally, RCNN and other models such as SSD are unable to accurately recognise tiny objects, but SSD is the fastest ObjectDetection Model at doing so.

## **RESULTS**

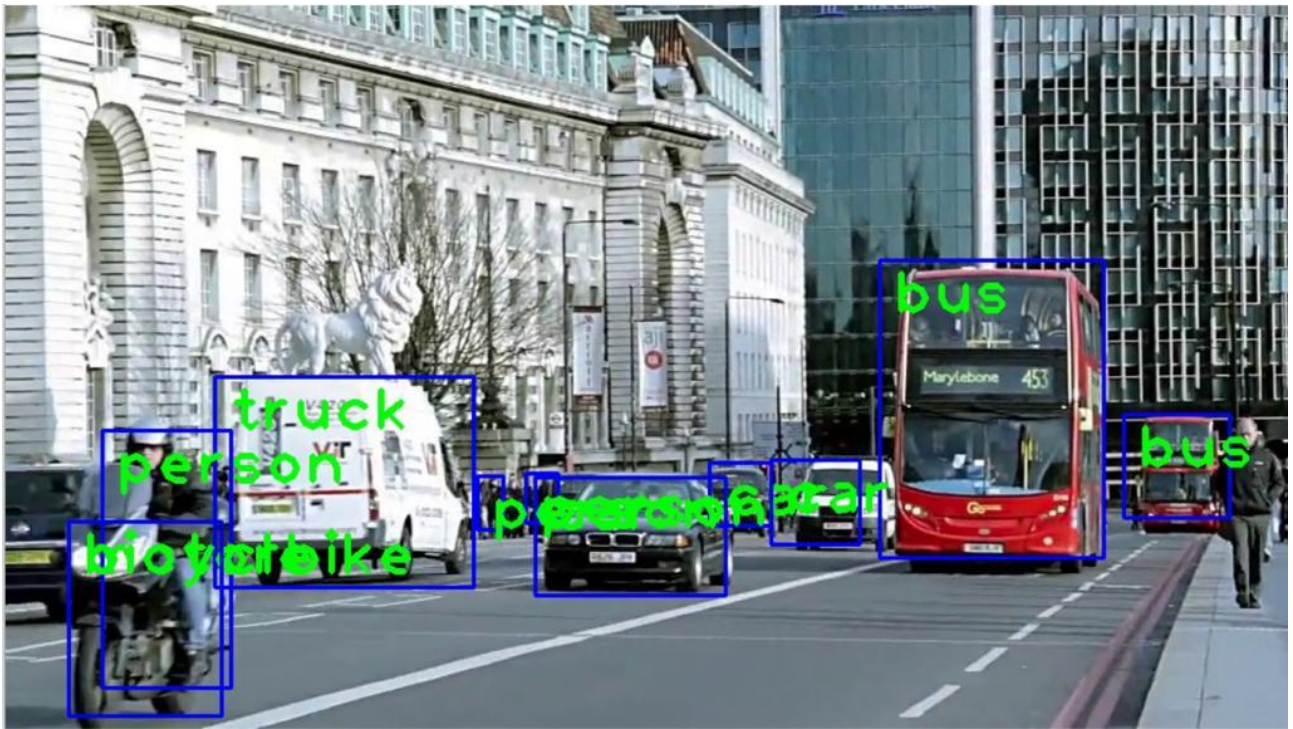


### **Figure – 13 Result of implementation**



### Figure – 14 Result of Implementation





**Figure – 15 Result of Implementation**



**Figure – 16 Result of Implementation**

## **CHAPTER – 07: CONCLUSION AND FUTURE WORK**

In this project, we are able to detect things the quickest by using SSD (single Shot Detector). It aids in the quickest identification of certain objects in the image.

We may detect numerous objects simultaneously in real time by utilising Single Shot Multi-Box Detector. We have demonstrated how Real-Time Object Detection systems can make use of the widely utilised fully convolutional neural networks.

In order to raise the accuracy of the objects recognised by raising the probability of the objects detected, the networks were trained on a low-cost GPU using the MSCOCO dataset. The performance of the model was then determined. Additionally, SSD outperformed Fast RCNN and Faster RCNN in object localisation.

Finally, we are able to perform Real-Time Object Detection using SSD with MobilNet v1 which is faster and efficient than traditional methods of object detection.

The objective of project is to develop an Object Recognition system to recognise the 2-D and 3-D items in the picture. The efficiency of the Object Detection systems can be further increased by boosting the global or local features. The suggested Object Detection method only employs greyscale photos, discarding the colour data. Since colour is so important in robotics, the colour information from the photos can be used to better correctly identify things. In addition, tracking devices and CCTV cameras can be equipped with a night vision mode as a built-in feature. Multiview tracking can be implemented utilising numerous cameras because of its vast coverage and various viewing angles for the objects to be tracked in order to make the systems entirely autonomous and to get over the aforementioned restrictions.

## **CHAPTER – 08: REFERENCES**

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. <https://www.geeksforgeeks.org/opencv-overview/>
5. 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
6. 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
7. Benbouzid, D., Busa-Fekete, R., and Kegl, B. (2012). “Fast classification using sparse decision dags,” in *Proceedings of the 29th International Conference on Machine Learning (ICML-12), ICML ‘12*, eds J. Langford and J. Pineau (New York, NY:Omnipress), 951–958.
8. Bengio, Y. (2012). “Deep learning of representations for unsupervised and transfer learning,” in *ICML Unsupervised and Transfer Learning, Volume 27 of JMLR Proceedings*, eds I. Guyon, G. Dror, V. Lemaire, G. W. Taylor, and D. L. Silver (Bellevue: JMLR.Org), 17–36.
9. 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.
10. Bourdev, L. D., and Malik, J. (2009). “Poselets: body part detectors trained using 3d human pose annotations,” in *IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 – October 4, 2009* (Kyoto: IEEE), 1365–1372.

11. 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).
12. 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.
13. Dalal, N., and Triggs, B. (2005). "Histograms of oriented gradients for human detection," in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, Vol. 1 (San Diego, CA: IEEE), 886–893. doi:10.1109/CVPR.2005.177.
14. Erhan, D., Szegedy, C., Toshev, A., and Anguelov, D. (2014). "Scalable object detection using deep neural networks," in Computer Vision and Pattern Recognition Frontiers in Robotics and AI [www.frontiersin.org](http://www.frontiersin.org) November 2015.

## **PERSONAL DETAILS**



**NAME :** Rangoli Jaiswal

**ER. NO.:** 191b199

**COURSE:** Bachelor in Technology

**BRANCH:** Computer Science and Engineering

**FATHER'S NAME:** Late Shrawan Kumar Jaiswal

**MOTHER'S NAME:** Mrs. Barsa Kumari

**EMAIL:** rangolijaiswal100@gmail.com

**MOBILE:** 9108257706





**NAME :** Rishi Neelkanth

**ER. NO.:** 191b201

**COURSE:** Bachelor in Technology

**BRANCH:** Computer Science and Engineering

**FATHER'S NAME:** Mr.Rajesh Kumar Srivastava

**MOTHER'S NAME:** Mrs.Rashmi Srivastava

**EMAIL:** rishi2001gkp@gmail.com

**MOBILE:** 7524980625



**NAME :** Sejal Jain

**ER. NO.:** 191B223

**COURSE:** Bachelor in Technology

**BRANCH:** Computer Science and Engineering

**FATHER'S NAME:** Mr.Shreyansh Jain

**MOTHER'S NAME:** Mrs.Baby Jain

**EMAIL:** jsejal970@gmail.com

**MOBILE:** 9610158575