**MACHINE LEARNING**

**(Object Detection)**

*Summer Internship Report Submitted in partial fulfillment*

*of the requirement for undergraduate degree of*

**Bachelor of Technology**

**In**

**Computer Science Engineering**

**By**

**SAI DURGA SRINIVASARAJU VETUKURI**

**221710305047**

*Under the Guidance of*

**Mr.**

Assistant Professor

****

Department Of Computer Science Engineering

GITAM School of Technology

GITAM (Deemed to be University) Hyderabad-502329 ,June 2019

i

DECLARATION

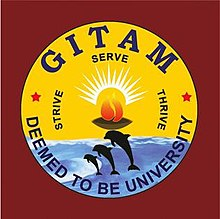
I submit this industrial training work entitled “OBJECT DETECTION” to GITAM (Deemed To Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of “**Bachelor of Technology**” in “**Computer Science Engineering**”. I declare that it was carried out independently by me under the guidance of **Mr.** , Asst. Professor, GITAM (Deemed To Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD SAI DURGA SRINIVASARAJU VETUKURI

Date: Roll No. 221710305047

ii

****

GITAM (DEEMED TO BE UNIVERSITY

Hyderabad-502329,India

Date:

**CERTIFICATE**

This is to certify that the Industrial Training Report entitled “OBJECT DETECTION” is being submitted by SAI DURGA SRINIVASARAJU VETUKURI (221710305047) in partial fulfillment of the requirement for the award of Bachelor of Technology in Computer Science Engineering at GITAM (Deemed To Be University), Hyderabad during the academic year 2020-21

It is faithful record work carried out by him at the Computer Science Engineering Department, GITAM University Hyderabad Campus under my guidance and supervision.

**Mr.**  **Dr.S.Phani Kumar**

Assistant Professor Professor and HOD

Department of CSE Department of CSE

iii

**iv**

**ACKNOWLEDGEMENT**

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this internship.

I would like to thank Dr. N. Siva Prasad, Pro Vice Chancellor, GITAM Hyderabad and Dr. N.Seetharamaiah, Principal, GITAM Hyderabad

I would like to thank respected Dr. S.PhaniKumar, Head of the Department of Computer Science Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present an internship report. It helped me a lot to realize of what we study for

I would like to thank the respected faculties Mr. who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end

SAI DURGA SRINIVASARAJU VETUKURI

Roll No. 221710305047

**v**

**ABSTRACT**

Machine learning algorithms are used to predict the values from the data set by splitting the data set in to train and test and building Machine learning algorithms models of higher accuracy to predict the values is the primary task to be performed on Cereals data set My perception of understanding the given data set has been in the view of undertaking a client’s requirement of overcoming the stagnant point of sales of the products being manufactured by client.

To get a better understanding and work on a strategic approach for solution of the client, I have adapted the viewpoint of looking at ratings of the products and for further deep understanding of the problem, I have taken the stance of a consumer and reasoned out the various factors of choice of the products and they purchase , and my primary objective of this case study was to look up the factors which were dampening the sale of products and corelate them to ratings of products and draft out an outcome report to client regarding the various accepts of a product manufacturing , marketing and sale point determination.

vi

**Table of Contents:**

**LIST OF FIGURES IX**

**CHAPTER 1:MACHINE LEARNING**

1.1 INTRODUCTION…………………………………………………………1

1.2 IMPORTANCE OF MACHINE LEARNING…………………………….1

1.3 USES OF MACHINE LEARNING……………………………………….2

1.4 TYPES OF LEARNING ALGORITHMS………………………………...3

1.4.1 Supervised Learning…………………………………………….3

1.4.2 Unsupervised Learning………………………………………….3

1.4.3 Semi Supervised Learning………………………………………4

1.5 RELATION BETWEEN DATA MINING, MACHINE LEARNING AND DEEP LEARNING……………………………………………………………………5

**CHAPTER 2:PYTHON**……...………………………………………………………6

2.1 INTRODUCTION TO PYTHON…………………………………………6

2.2 HISTORY OF PYTHON………………………………………………….6

2.3 FEATURES OF PYTHON………………………………………………..6

2.4 HOW TO SETUP PYTHON……………………………………………...7

2.4.1 Installation(using python IDLE)………………………………..7

2.4.2 Installation(using Anaconda)…………………………………...7

2.5 PYTHON VARIABLE TYPES…………………………………………...9

2.5.1 Python Numbers………………………………………………...9

2.5.2 Python Strings…………………………………………………..9

2.5.3 Python Lists…………………………………………………….9

2.5.4 Python Tuples………………………………………………….10

2.5.5 Python Dictionary……………………………………………..10

2.6 PYTHON FUNCTION…………………………………………………..10

2.6.1 Defining a Function…………………………………………...10

2.6.2 Calling a Function……………………………………………..11

vii

2.7 PYTHON USING OOPs CONCEPTS…………………………………..11

2.7.1 Class…………………………………………………………...11

2.7.2 \_\_init\_\_method in class……………………………………….11

**CHAPTER 3:CASE STUDY..……………………………………………………..12**

3.1 PROBLEM STATEMENT………………………………………………12

3.2 DATA SET………………………………………………………………12

3.3 OBJECTIVE OF THE CASE STUDY…………………………………..12

**CHAPTER 4:MODEL BUILDING…..…………………………………………….13**

4.1 PRE PROCESSING OF DATA(Local System)….……………………....13

4.1.1 Getting the Dataset……………………………………………..13

4.1.2 Download Tensorflow Models…………………………………17

4.1.3 Set up Anaconda Virtual Environment and install packages......18

4.1.4 Convert XML files to CSV……….…………………………….20

4.2 PRE PROCESSING OF DATA(Google Colab)….…………………….....22

4.2.1 Installing Tensorflow in Google Colab..………………………..23

4.2.2 Import Tensorflow and Chek for GPU………………………….23

4.2.3 Mount Google Drive……………………………………….....…24

4.2.4 Install Some Tools and Dependencies....………………………..24

4.2.5 Create TF Records for Train and Test Data…………………….26

**CHAPTER 5: TRAIN THE OBJECT DETECTION MODEL…………………..30**

5.1 OPEN TENSORBOARD………………………………………………….30

5.2 CREATE LABELMAP.PBTXT FILE….………………………………...30

5.3 EDIT YOUR MODEL CONFIG FILE…………………………………...31

5.4 TRAIN YOUR MODEL………………………………………………….35

**CHAPTER 6: TEST YOUR MODEL……………………………………………...38**

6.1 EXPORT THE TRAIN MODEL..………………………………………..38

viii

6.2 RUN THE OBJECT DETECTION MODEL..…………………………...39

**CHAPTER 7: OBJECT DETECTION………………...…………………………..44**

**CONCLUSION………………………………………………………………………48**

**REFERENCES……………………………………………………………………....49**

ix

**LIST OF FIGURES:**

Figure 1 : The Process Flow………………………………………….………..2

Figure 2 : Unsupervised Learning……………………………………………..4

Figure 3 : Semi Supervised Learning………………………………………….4

Figure 4 : Python Download…………………………………………………...7

Figure 5 : Anaconda Download………………………………………………..8

Figure 6 : Jupyter Notebook…………………………………………………...8

Figure 7 : Change directory to OIDV4\_Toolkit...…………………………….13

Figure 8 : Image Dataset………………………………………………………15

Figure 9 : Labeling of image……..………………………………...……...….16

Figure 10 : After Spliting train and test images…………………………...….16

Figure 11 : Tensorflow Models Directory..………………………………...…17

Figure 12 : Anaconda Prompt…………………………………………………18

Figure 13 : Output of xml\_to\_csv.py………………………………………….21

Figure 14 : Label csv files in image directory...………………………………21

Figure 15 : Upload models folder in Google Drive.…………………………..22

Figure 16 : Set the Hardware accelerator as GPU..…………………………...22

Figure 17 : Select the Tensorflow version 1.x..……………………………….23

Figure 18 : Check GPU is enabled or not...……………………………………23

Figure 19 : Mount Google Drive..……………………………………………..24

Figure 20 : Change directory to Car\_And\_Number\_Plate..…………………...24

Figure 21 : Install the dependencies..………………………………………….24

Figure 22 : Compile protobuf files…………………………………………….25

Figure 23 : Create Python Path………………………………………………...25

Figure 24 : Run for every session restart………………………………………25

Figure 25 : Check remaining session time.…………………………………….25

Figure 26 : Check whether all requirements for training is installed...……….26

Figure 27 : Change directory to object\_detection……………………………..26

Figure 28 : Creation of TF records…………………………………………….29

Figure 29 : Load the Tensorboard…………………………………………….30

Figure 30 : Reload the Tensorboard…………………………………………..30

Figure 31 : Output after training……………………………………………...35

Figure 32 : Graph of Learning Rate…………………………………………..36

Figure 33 : Graph of Total Loss………………………………………………37

Figure 34 : Checkpoints in the training directory…………………………….38

Figure 35 : frozen\_inference\_graph.pb in inference\_graph directory………..39

Figure 36 : Output(1)………………………………………………………….43

Figure 37 : Output(2)………………………………………………………….43

Figure 38 : Faster R-CNN(1)………………………………………………….44

Figure 39 : Faster R-CNN(2)………………………………………………….45

Figure 40 : SSD(1)…………………………………………………………….45

Figure 41 : YOLO(1)………………………………………………………….46

Figure 42 : YOLO(2)………………………………………………………….47

**CHAPTER 1**

**MACHINE LEARNING**

**1.1 INTRODUCTION:**

Machine Learning(ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of Artificial Intelligence(AI).

**1.2 IMPORTANCE OF MACHINE LEARNING:**

Consider some of the instances where machine learning is applied: the self-driving Google car, cyber fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and “more items to consider” and “get yourself a little something” on Amazon—are all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take in today’s data-rich world.

Machines can aid in filtering useful pieces of information that help in major advancements, and we are already seeing how this technology is being implemented in a wide variety of industries.

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that’s in part due to increased sophistication of machine learning, which helps analyze those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques.

The process flow depicted here represents how machine learning works

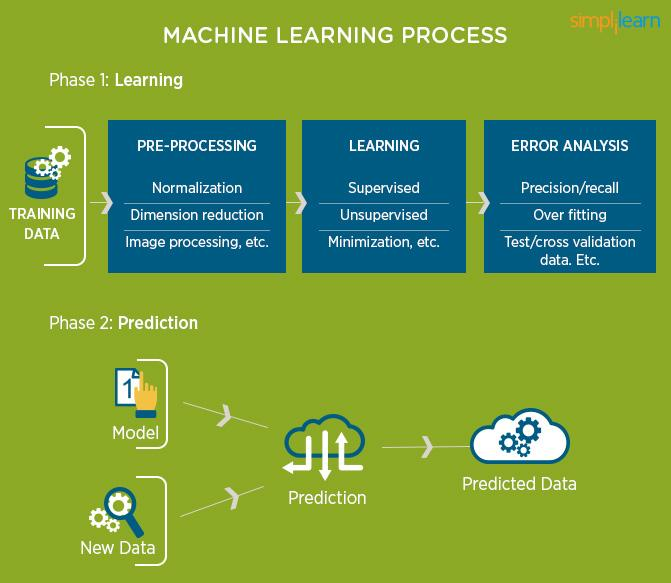


Figure 1 : The Process Flow

**1.3 USES OF MACHINE LEARNING:**

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let’s consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyze huge volumes of data .

Traditionally, data analysis was always characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives to analyzing huge volumes of data.

By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning can produce accurate results and analysis.

**1.4 TYPES OF LEARNING ALGORITHMS:**

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

**1.4.1 Supervised Learning :**

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of supervised learning.

Supervised machine learning algorithms uncover insights, patterns, and relationships from a labelled training dataset – that is, a dataset that already contains a known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem during training, it is able to “learn” how the rest of the features relate to the target, enabling you to uncover insights and make predictions about future outcomes based on historical data.

Examples of Supervised Machine Learning Techniques are Regression, in which the algorithm returns a numerical target for each example, such as how much revenue will be generated from a new marketing campaign.

Classification, in which the algorithm attempts to label each example by choosing between two or more different classes. Choosing between two classes is called binary classification, such as determining whether or not someone will default on a loan. Choosing between more than two classes is referred to as multiclass classification.

**1.4.2 Unsupervised Learning:**

When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

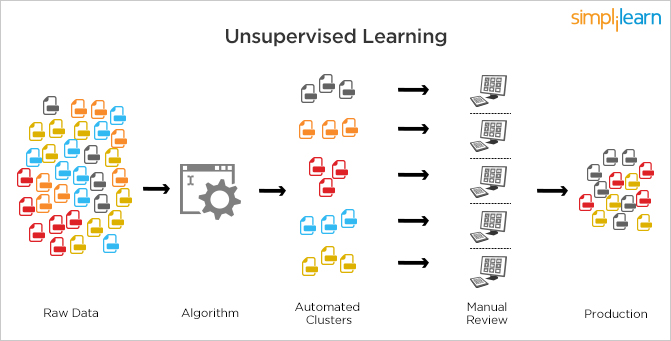


Figure 2 : Unsupervised Learning

Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbor mapping, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning.

**1.4.3 Semi Supervised Learning:**

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labeled and unlabeled data for training. In a typical scenario, the algorithm would use a small amount of labeled data with a large amount of unlabeled data.

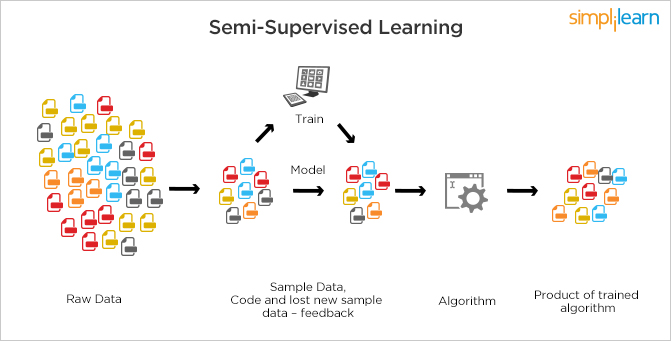


Figure 3 : Semi Supervised Learning

**1.5 RELATION BETWEEN DATA MINING,MACHINE LEARNING AND DEEP LEARNING:**

Machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining discovered previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge—and further automatically applies that information to data, decision-making, and actions.

Deep learning, on the other hand, uses advanced computing power and special 5 types of neural networks and applies them to large amounts of data to learn, understand, and identify complicated patterns. Automatic language translation and medical diagnoses are examples of deep learning.

**CHAPTER 2**

**PYTHON**

Basic programming language used for machine learning is : PYTHON

**2.1 INTRODUCTION TO PYTHON:**

**●** Python is a high-level, interpreted, interactive and object-oriented scripting language.

● Python is a general purpose programming language that is often applied in scripting roles

● Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.

● Python is Interactive: You can sit at a Python prompt and interact with the interpreter directly to write your programs.

● Python is Object-Oriented: Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.

**2.2 HISTORY OF PYTHON:**

● Python was developed by GUIDO VAN ROSSUM in early 1990’s

● Its latest version is 3.7 , it is generally called as python3

**2.3 FEATURES OF PYTHON:**

● Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax, This allows the student to pick up the language quickly.

● Easy-to-read: Python code is more clearly defined and visible to the eyes.

● Easy-to-maintain: Python's source code is fairly easy-to-maintaining.

● A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

● Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

● Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

● Databases: Python provides interfaces to all major commercial databases.

● GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

**2.4 HOW TO SETUP PYTHON:**

● Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

● The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python.

**2.4.1 Installation(using python IDLE):**

● Installing python is generally easy, and nowadays many Linux and Mac OS distributions include a recent python.

● Download python from www.python.org

● When the download is completed, double click the file and follow the instructions to install it.

● When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.

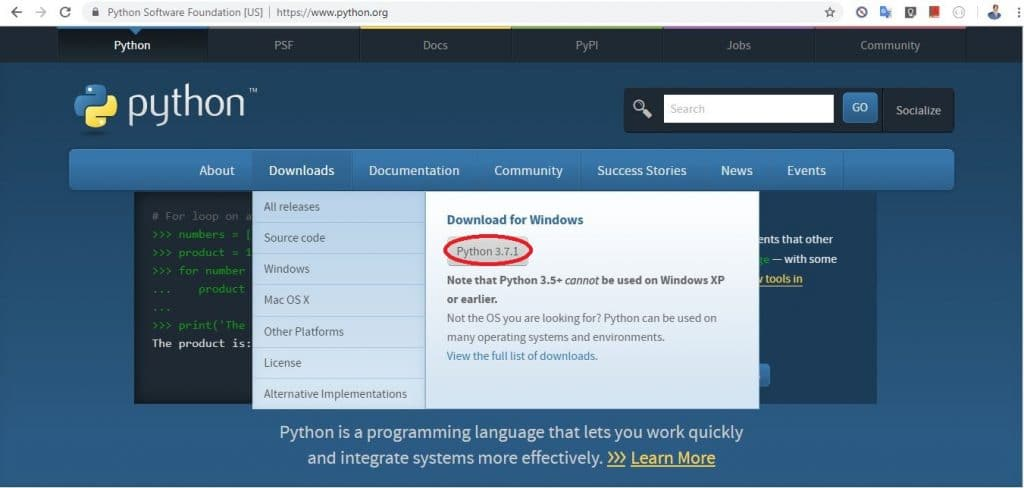


Figure 4 : Python download

**2.4.2 Installation(using Anaconda):**

● Python programs are also executed using Anaconda.

● Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.

● Conda is a package manager quickly installs and manages packages.

● In WINDOWS:

● In windows

● Step 1: Open Anaconda.com/downloads in a web browser.

● Step 2: Download python 3.4 version for (32-bits graphic installer/64 -bit graphic installer)

● Step 3: select installation type( all users)

● Step 4: Select path(i.e. add anaconda to path & register anaconda as default python 3.4) next click install and next click finish

● Step 5: Open jupyter notebook ( it opens in default browser)

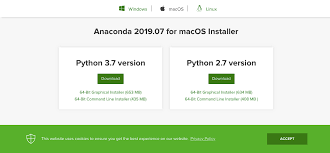


Figure 5 : Anaconda download

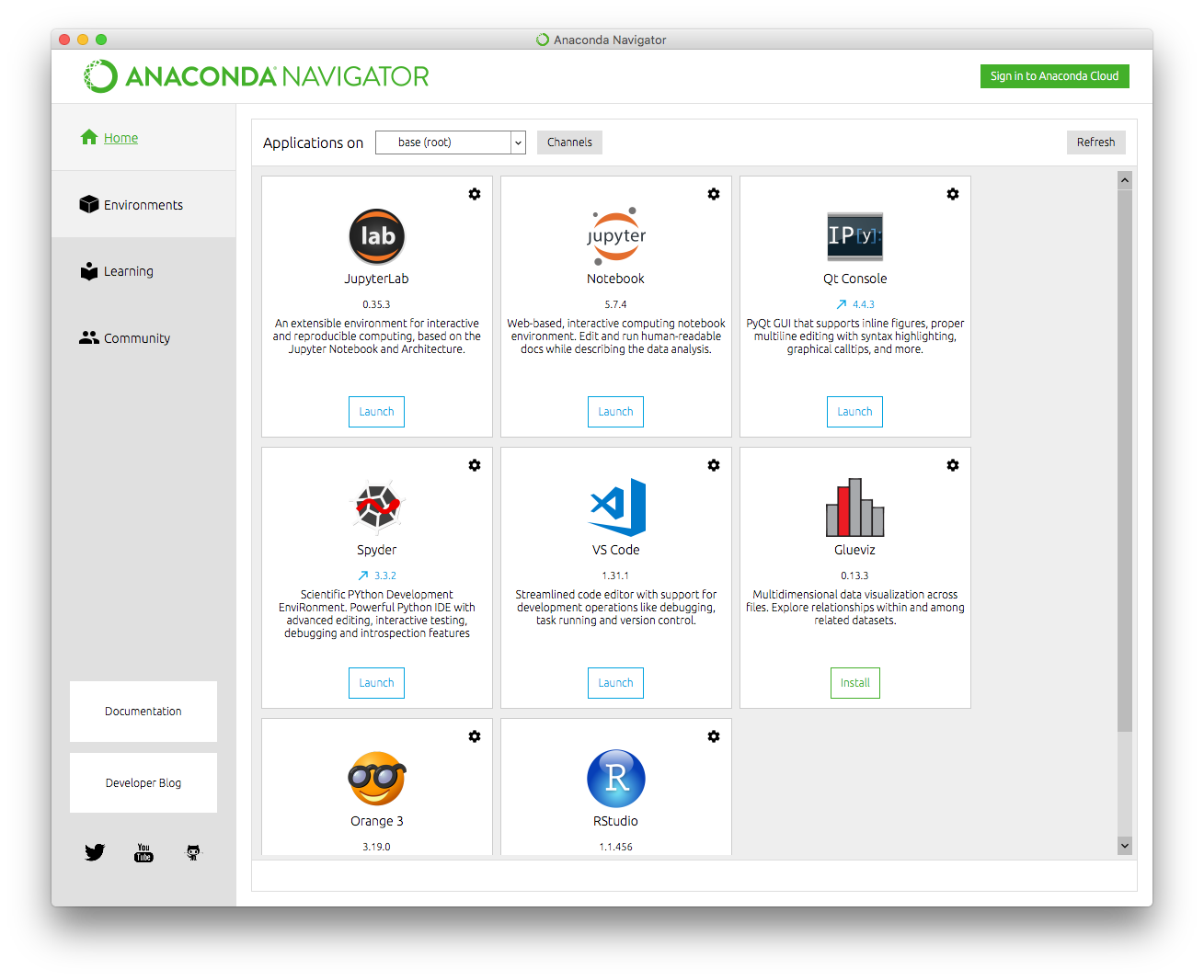


Figure 6 : Jupyter notebook

**2.5 PYTHON VARIABLE TYPES:**

● Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

● Variables are nothing but reserved memory locations to store values.

● Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.

● Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.

● Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

● Python has five standard data types –

o Numbers

o Strings

o Lists

o Tuples

o Dictionary

**2.5.1 Python Numbers:**

● Number data types store numeric values. Number objects are created when you assign a value to them.

● Python supports four different numerical types − int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

**2.5.2 Python Strings:**

● Strings in Python are identified as a contiguous set of characters represented in the quotation marks.

● Python allows for either pairs of single or double quotes.

● Subsets of strings can be taken using the slice operator ([ ] and [:] ) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

● The plus (+) sign is the string concatenation operator and the asterisk (\*) is the repetition operator.

**2.5.3 Python Lists:**

● Lists are the most versatile of Python's compound data types.

● A list contains items separated by commas and enclosed within square brackets ([]).

● To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

● The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.

● The plus (+) sign is the list concatenation operator, and the asterisk (\*) is the repetition operator.

**2.5.4 Python Tuples:**

● A tuple is another sequence data type that is similar to the list.

● A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

● The main differences between lists and tuples are: Lists are enclosed in brackets ( [ ] ) and their elements and size can be changed, while tuples are enclosed in parentheses ( ( ) ) and cannot be updated.

● Tuples can be thought of as read-only lists.

● For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a tuple. Tuples have no remove or pop method.

**2.5.5 Python Dictionary:**

● Python's dictionaries are a kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

● Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

● You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.

● What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

**2.6 PYTHON FUNCTION:**

**2.6.1 Defining a Function:**

You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function name and parentheses (i.e.()).

Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses

The code block within every function starts with a colon (:) and is indented. The statement returns [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

**2.6.2 Calling a Function:**

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

**2.7 PYTHON USING OOP’s CONCEPTS:**

**2.7.1 Class:**

● Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.

● Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.

● Data member: A class variable or instance variable that holds data associated with a class and its objects.

● Instance variable: A variable that is defined inside a method and belongs only to the current instance of a class.

● Defining a Class:

o We define a class in a very similar way how we define a function.

o Just like a function ,we use parentheses and a colon after the class name(i.e. ():) when we define a class. Similarly, the body of our class is 14 indented like a functions body is indented like a functions body is.

**2.7.2 \_\_init\_\_ method in Class:**

● The init method — also called a constructor — is a special method that runs when an instance is created so we can perform any tasks to set up the instance.

● The init method has a special name that starts and ends with two underscores:\_\_init\_\_().

**CHAPTER 3**

**CASE STUDY**

**3.1 PROBLEM STATEMENT:**

To recognize car and vehicle registration plate object in an image

**3.2 DATA SET:**

* Collect the images from the Open Image V6 for Car and vehicle registration plate. For each image there must be annotations present for objects to be detected.
* There are separate annotation files for each image. The Annotations are the xml files. All Annotation XML data must be included in one csv files.
* There are 2 csv files used for training the object detection model i.e. train\_labels.csv and test\_labels.csv.

Both train\_labels.csv and test\_labels.csv consist of the following parameters:

1. **Filename-** It is the column for the image filename with respective path.
2. **Width-** It is the column for the width of the particular image.
3. **Height-**It is the column for height of the particular image.
4. **Class-**It is the column for telling the object class present in the image (particular object image).
5. **Points of the Single Class Object-**The points for the single class object are given as **Xmin, Ymin, Xmax and Ymax.**These points represent the location of the object in the image.

**3.3 OBJECTIVE OF THE CASE STUDY:**

To detect objects in the image based on the trained image data set by using Tensorflow and Deep Learning (Convolutional Neural networks).This model will help the traffic police to identify the vehicle owner who is breaking the traffic rules through the Vehicle registration plate details. As the Vehicle registration number is visible it will detect the Vehicle registration plate present on the vehicle.

**CHAPTER 4**

**MODEL BUILDING**

**4.1 PREPROCESSING OF THE DATA (Local System):**

Preprocessing of the data in local system before going to colab actually involves the following steps:

**4.1.1 GETTING THE DATASET:**

Get the images for the respective object required from Open Image V6

<https://storage.googleapis.com/openimages/web/index.html> (To check the images for object you required click on explore and then search for the images)

Download all the images you required by following the steps below:

1. Download the OIDv4\_Toolkit from this link:

<https://github.com/theAIGuysCode/OIDv4_ToolKit>

1. Put the downloaded directory in your own folder
2. Change the directory to the directory which contains OIDV4\_ToolKit directory in command prompt.

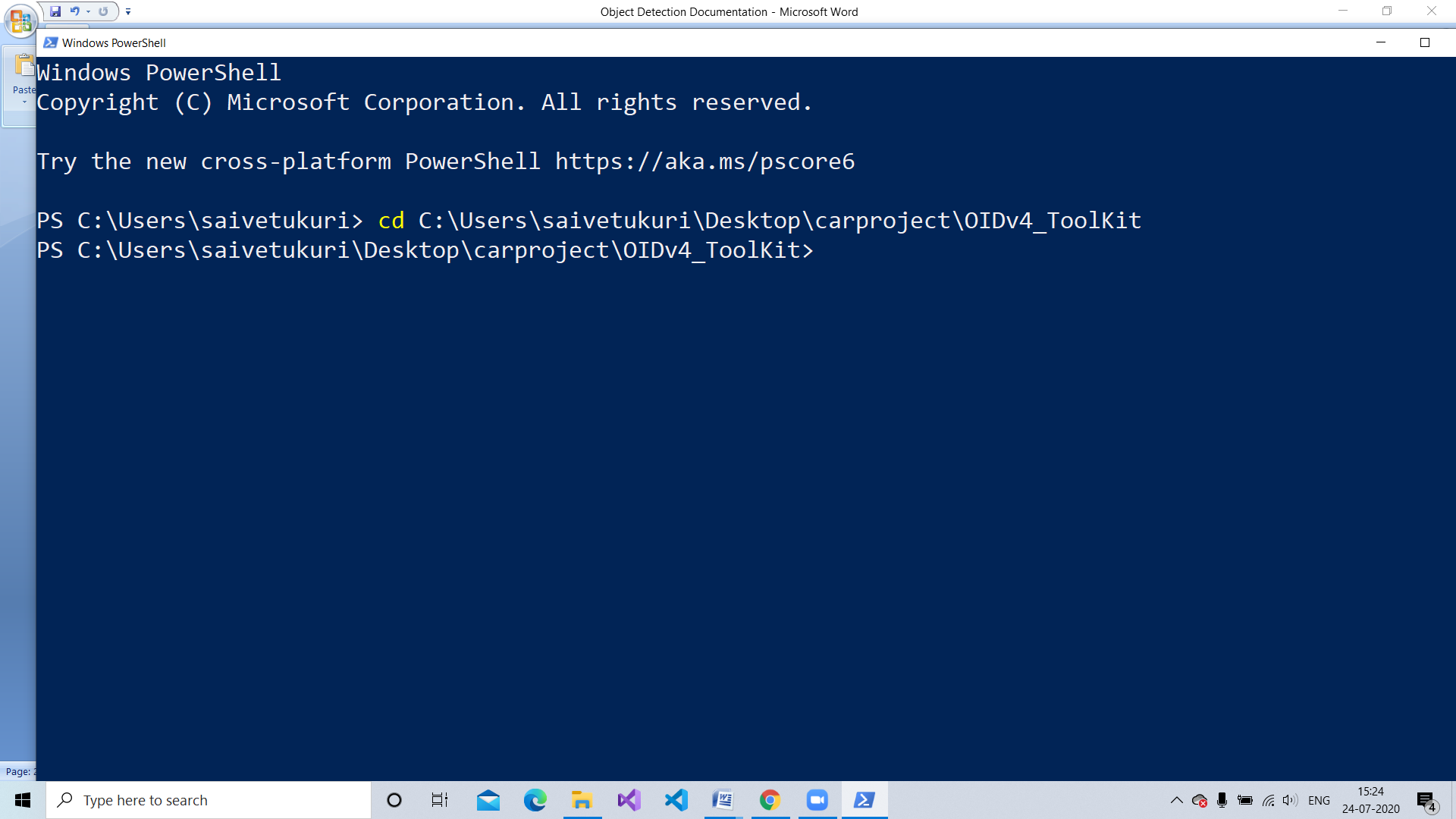


Figure 7 : Change directory to OIDV4\_Toolkit

1. Make sure you install all the packages in OIDV4\_ToolKit directory you have downloaded.

The required packages are as follows:

* pandas
* numpy
* awscli
* urllib3
* tqdm
* opencv-python

All the required packages are present in requirements.txt, So type the following :

|  |
| --- |
| pip install -r requirements.txt |

1. To Download the required images based on the required object class execute main.py present in OIDv4\_ToolKit directory as follows:

|  |
| --- |
| python main.py downloader --classes Car "Vehicle registration plate" --type\_csv train --limit 1000 --multiclass 1 |

In this command 1000 images is given as the limit for each class[“Car” and “Vehicle registration plate”].If the images present is less than the limit then it will download the all images present.

**main.py**

|  |
| --- |
| **from** sys **import** exit **from** textwrap **import** dedent **from** modules.parser **import** \* **from** modules.utils **import** \* **from** modules.downloader **import** \* **from** modules.show **import** \* **from** modules.csv\_downloader **import** \* **from** modules.bounding\_boxes **import** \* **from** modules.image\_level **import** \*   ROOT\_DIR = **''** DEFAULT\_OID\_DIR = os.path.join(ROOT\_DIR, **'OID'**)  **if** \_\_name\_\_ == **'\_\_main\_\_'**:   args = parser\_arguments()   **if** args.command == **'downloader\_ill'**:  image\_level(args, DEFAULT\_OID\_DIR)  **else**:  bounding\_boxes\_images(args, DEFAULT\_OID\_DIR) |

1. After downloading is finished the images will be present in the following directory:

|  |
| --- |
| OIDv4\_ToolKit\OID\Dataset\train\ |

Based the classes you asked for downloaded the folder name is given .

The images consist of “images” and “Label folder”

For Example:

For Car and Vehicle Registration Plate images it is “Car\_Vehicle registration plate”

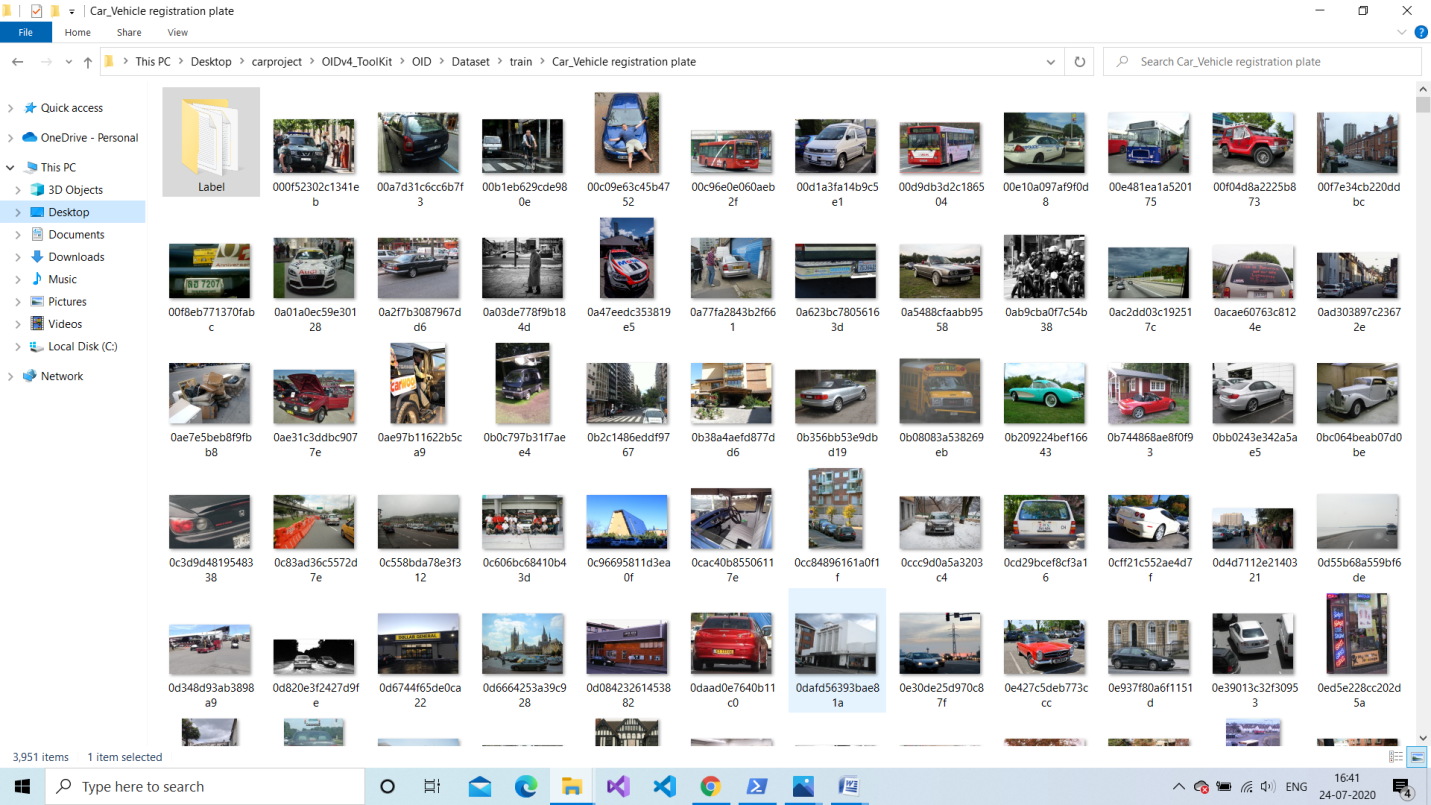


Figure 8 : Image Dataset

Labels folder consist of txt files for respective image. I prefer to have in xml so I make my own labels by using “LabelImg” which is explained in the next step.

**Note:** Txt files is used in YOLO model but we are using faster r-cnn model for my project so we require xml files.

1. As we xml files for our model ,Let us label our images and name the class for the particular marked region.

LabelImg is the software used to label my images .



Using the Software mark region which consist the object you required and name it. Do it for each and every image you have. An XML file is produced for respective images.



Figure 9 : Labeling of image

In XML file the required information to take is:

* Image filename
* Width of the image
* Height of the image
* Object name present in the images with the points Xmin,Ymin,Xmax and Ymax

1. Separate the images to train and test by creating folders for it.

**Note:** Make sure that train consist 80% to 90% of images data and test with 10% to 20%.

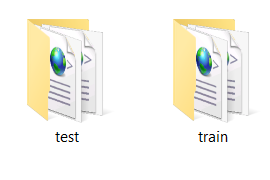


Figure 10 : After Spliting train and test images

**Note:** Make sure the image file size is below 250kb so that the training will go faster.

* + 1. **DOWNLOAD TENSORFLOW MODELS:**

1. From the following link download the models folder of tensorflow.

<https://github.com/tensorflow/models>

Extract the zip folder and place it into your project directory

Models Folder consist of the following files and folders:

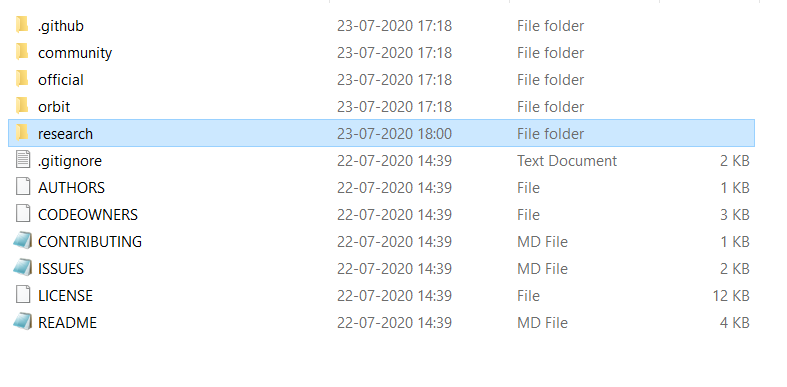


Figure 11 : Tensorflow Models Directory

1. Download the required object detection model from the following link:

<https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md>

As we are using Faster rcnn model in our project.

So, download “faster\_rcnn\_inception\_v2\_coco\_2018\_01\_28 .tar.gz”.

Extract the file and place it into the directory “\models\research\object\_detection”

1. Go to the directory “\models\research\object\_detection”.
2. Create the folders as follows:
   1. Images- Which have both test and train folders.
   2. interface\_graph- Which is used to store the frozen\_interface\_graph.pb that will be used for object detection in the images and videos.
   3. Training- To store all checkpoints while training the model and consist of the labelmap.pbtxt which will be used to name the detected object.
3. Create the files as follows:
   1. xml\_to\_csv.py – This will convert all the train xml files and test xml files into the one csv files as train\_labels.csv and test\_labels.csv respectively.
   2. generate\_tfrecord.py – this is to create the records for the train and test data based on the train\_labels.csv and test\_labels.csv .Finally gives train.records and test.records.
      1. **SET UP ANACONDA VIRTUAL ENVIRONMENT AND INSTALL PACKAGES:**
4. Open anaconda prompt as administrator.

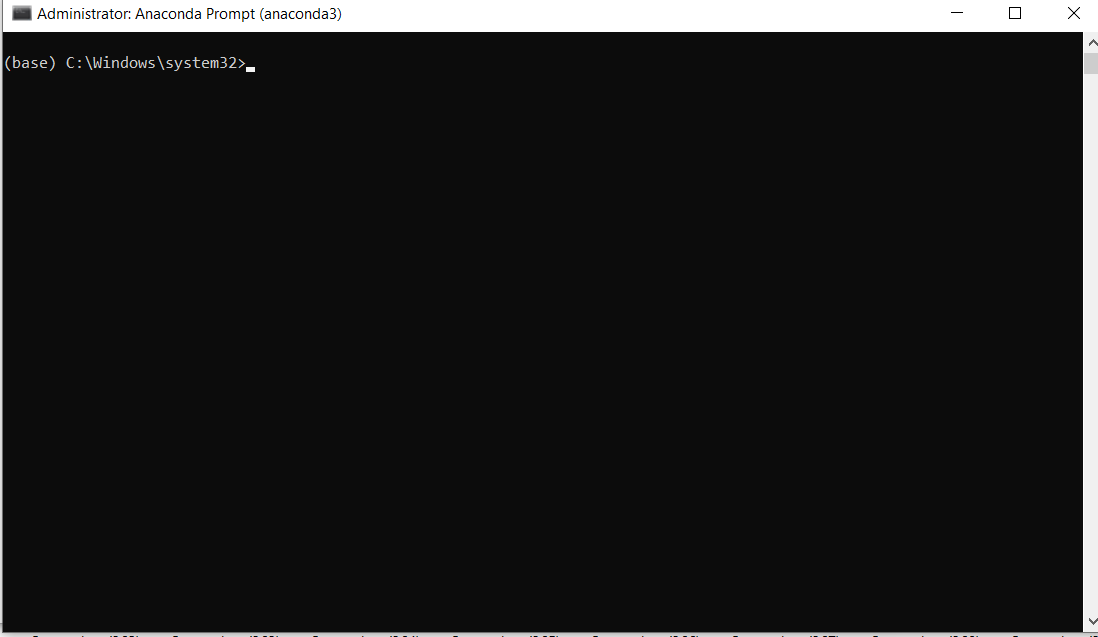


Figure 12: Anaconda prompt

In the Anaconda prompt type the command given below from step 2 to

1. Create your own virtual environment by typing

|  |
| --- |
| conda create –n car pip |

1. After the creating your virtual environment , activate the virtual environment you created. Type the following command

|  |
| --- |
| activate car |

1. Install the package tensorflow-gpu in this environment you created

|  |
| --- |
| pip install --ignore-installed --upgrade tensorflow-gpu |

1. Install the other necessary packages by typing the following commands one by one:

|  |
| --- |
| conda install -c anaconda protobuf  pip install pillow  pip install lxml  pip install jupyter  pip install matplotlib  pip install pandas  pip install opencv-python |

1. As all packages required are installed. Now we need to add some folders in the environmental variables

|  |
| --- |
| set PYTHONPATH = C:\project-directory\models; C:\project-directory\models\research; C:\project-directory\models\research\slim |

1. Add the PYTHONPATH variable to the PATH VARIABLE

|  |
| --- |
| set PATH= %PATH%;PYTHONPATH |

1. To check whether it is added to the path type the following

|  |
| --- |
| echo %PATH% |

1. Compile the protobuf files,which are used by Tensorflow to configure model and training parameters

Before we need to change directory to project-directory\models\research

|  |
| --- |
| cd C:\project-directory\models\research |

To compile the protobuf files type the following:

|  |
| --- |
| protoc --python\_out=. .\object\_detection\protos\anchor\_generator.proto .\object\_detection\protos\argmax\_matcher.proto .\object\_detection\protos\bipartite\_matcher.proto .\object\_detection\protos\box\_coder.proto .\object\_detection\protos\box\_predictor.proto .\object\_detection\protos\eval.proto .\object\_detection\protos\faster\_rcnn.proto .\object\_detection\protos\faster\_rcnn\_box\_coder.proto .\object\_detection\protos\grid\_anchor\_generator.proto .\object\_detection\protos\hyperparams.proto .\object\_detection\protos\image\_resizer.proto .\object\_detection\protos\input\_reader.proto .\object\_detection\protos\losses.proto .\object\_detection\protos\matcher.proto .\object\_detection\protos\mean\_stddev\_box\_coder.proto .\object\_detection\protos\model.proto .\object\_detection\protos\optimizer.proto .\object\_detection\protos\pipeline.proto .\object\_detection\protos\post\_processing.proto .\object\_detection\protos\preprocessor.proto .\object\_detection\protos\region\_similarity\_calculator.proto .\object\_detection\protos\square\_box\_coder.proto .\object\_detection\protos\ssd.proto .\object\_detection\protos\ssd\_anchor\_generator.proto .\object\_detection\protos\string\_int\_label\_map.proto .\object\_detection\protos\train.proto .\object\_detection\protos\keypoint\_box\_coder.proto .\object\_detection\protos\multiscale\_anchor\_generator.proto .\object\_detection\protos\graph\_rewriter.proto .\object\_detection\protos\calibration.proto .\object\_detection\protos\flexible\_grid\_anchor\_generator.proto |

Now each pb2 file in directory models/research/object\_detection/protos there is a protobuf file created.

1. To make the tensorflow object detection api set to use the pretrained models for object detection or create a new one, we need to type the following commands:

|  |
| --- |
| python setup.py build  python setup.py install |

**4.1.4 CONVERT XML FILES TO CSV:**

Running the xml\_to\_csv.py we will convert all the train and test xml files into train\_labels.csv and test\_labels.csv

xml\_to\_csv.py

|  |
| --- |
| **import** os **import** glob **import** pandas **as** pd **import** xml.etree.ElementTree **as** ET   **def** xml\_to\_csv(path):  xml\_list = []  **for** xml\_file **in** glob.glob(path + **'/\*.xml'**):  tree = ET.parse(xml\_file)  root = tree.getroot()  **for** member **in** root.findall(**'object'**):  value = (root.find(**'filename'**).text,  int(root.find(**'size'**)[0].text),  int(root.find(**'size'**)[1].text),  member[0].text,  int(member[4][0].text),  int(member[4][1].text),  int(member[4][2].text),  int(member[4][3].text)  )  xml\_list.append(value)  column\_name = [**'filename'**, **'width'**, **'height'**, **'class'**, **'xmin'**, **'ymin'**, **'xmax'**, **'ymax'**]  xml\_df = pd.DataFrame(xml\_list, columns=column\_name)  **return** xml\_df   **def** main():  **for** folder **in** [**'train'**,**'test'**]:  image\_path = os.path.join(os.getcwd(), (**'images/'** + folder))  xml\_df = xml\_to\_csv(image\_path)  xml\_df.to\_csv((**'images/'** + folder + **'\_labels.csv'**), index=**None**)  print(**'Successfully converted xml to csv.'**)   main() |

Change the directory to C:\project-directory\models\research\object\_detection

|  |
| --- |
| cd C:\project-directory\models\research\object\_detection |

For converting the xml to csv execute the xml\_to\_csv.py

|  |
| --- |
| python xml\_to\_csv.py |



Figure 13 : Output of xml\_to\_csv.py

If this above output appear it tells that the xml files are converted to train\_labels.csv and test\_labels.csv in the \models\research\object\_detection\images directory where the test and train images are present



Figure 14: Label csv files in images directory

**4.2 PREPROCESSING OF THE DATA(Google Colab):**

As soon as the preprocessing of data in the local system is done upload the models folder in a project directory you create in the Google Drive.

After Uploading the models in the Project directory

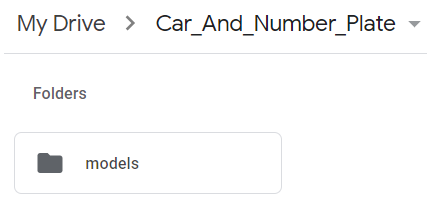


Figure 15 : Upload models folder in Google Drive

Create a new Google Colab Notebook to undergo remaining preprocessing work, training and testing of the model.

Before running the cells in colab make sure your notebook using “GPU”

For making your notebook use GPU ,

Click on Edit then Notebook settings

Change the hardware accelerator from None to GPU

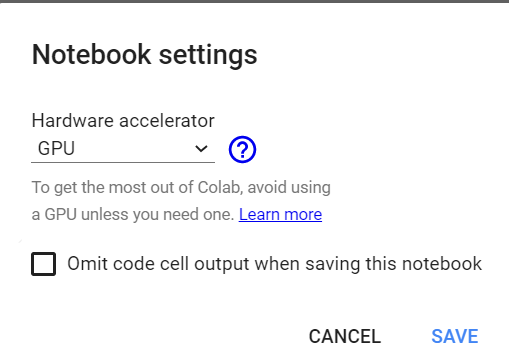


Figure 16 : Set the Hardware accelerator as GPU

Preprocessing of the data actually involves the following steps:

**4.2.1 INSTALL TENSORFLOW IN COLAB :**

Due to the upgrade in the TensorFlow on colab, run the code below. Since object detection API for TensorFlow, 2.0 hasn't been updated as of the time this publication is been reviewed.

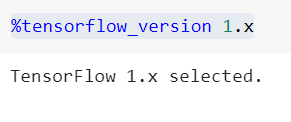


Figure 17 : Select the Tensorflow version 1.x

**4.2.2 IMPORT TENSORFLOW AND CHECK FOR GPU:**

Import the package tensorflow as we will be using that package for the whole project.

**Tensorflow:** TensorFlow is an open source machine learning framework for all developers. It is used for implementing machine learning and deep learning applications. To develop and research on fascinating ideas on artificial intelligence, Google team created TensorFlow. TensorFlow is designed in Python programming language, hence it is considered an easy to understand framework.

Check whether the GPU is enabled or not, By using GPU we will get the training of our model faster compared with CPU.

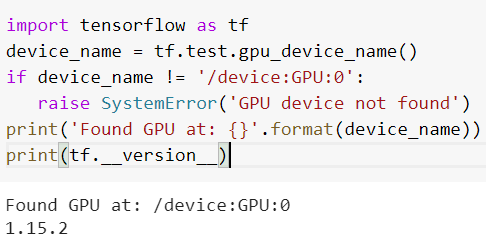


Figure 18 : Check GPU is enabled or not

If the GPU is enabled then we get the output “Found GPU at: /device:GPU:0” or else it gives an error as “GPU device not found”.

**4.2.3 MOUNT GOOGLE DRIVE:** Mounting the Google Drive we make you to access the files in the google drive for the respective account you use. It will ask the authorization for access your files before mounting. Once the Authorization is confirmed then Google Drive is Mounted.



Figure 19 : Mount Google Drive

**4.2.4 INSTALL SOME TOOLS AND DEPENDENCIES:**

In the project directory install the following dependencies:

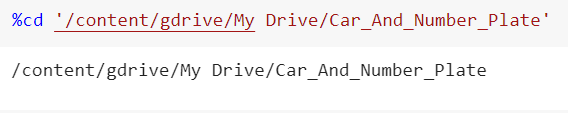


Figure 20 : Change directory to Car\_And\_Number\_Plate

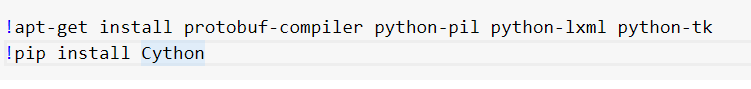


Figure 21: Install the dependencies

Compile the protobuf files in the research folder:

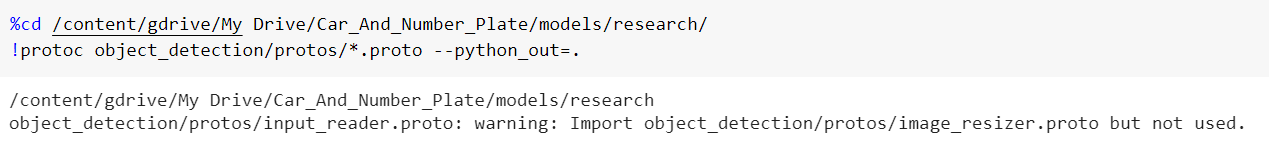


Figure 22 : Compile protobuf files

Create an Python Path for setting the environment:

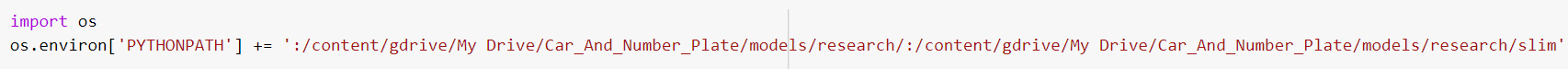


Figure 23: Create Python Path

Always run the below code for every session restart in the colab:

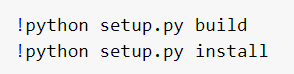


Figure 24: Run for every session restart

To Know the Remaining Session time in Colab Run the following code:

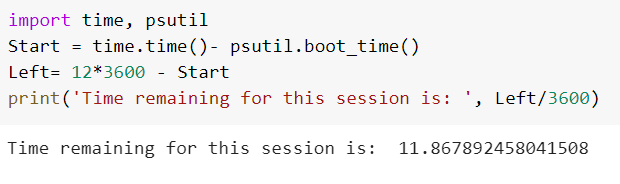


Figure 25 : Check remaining session time

Finally check whether all we need for the training has been installed:

1. Install the tf.slim package to run the model\_builder\_test.py code that is present in the directory “<http://content/gdrive/My%20Drive/Car_And_Number_Plate/models/research/object_detection/builders/>”
2. Run the model\_builder\_test.py program

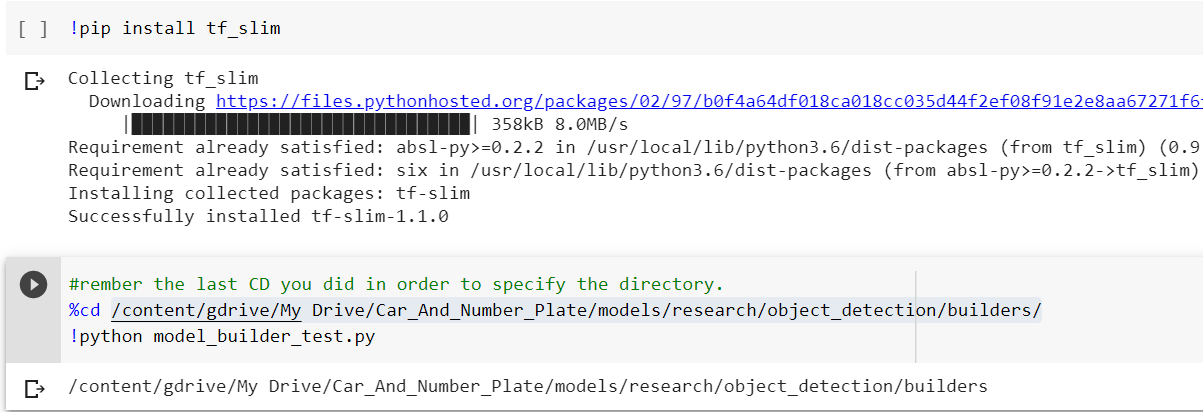


Figure 26 : Check whether all requirements for training is installed

**4.2.5 CREATE TF RECORDS FOR TRAIN AND TEST DATA:**

Run the generate\_tfrecord.py by mentioning the Label csv file names and image directory for the train and test separately.

Change your directory to /object\_detection :

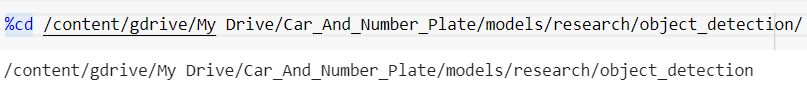


Figure 27 : Change directory to object\_detection

**generate\_tfrecords.py**

|  |
| --- |
| from \_\_future\_\_ import division  from \_\_future\_\_ import print\_function  from \_\_future\_\_ import absolute\_import  import os  import io  import pandas as pd  import tensorflow as tf  from PIL import Image  from object\_detection.utils import dataset\_util  from collections import namedtuple, OrderedDict  flags = tf.app.flags  flags.DEFINE\_string('csv\_input', '', 'Path to the CSV input')  flags.DEFINE\_string('image\_dir', '', 'Path to the image directory')  flags.DEFINE\_string('output\_path', '', 'Path to output TFRecord')  FLAGS = flags.FLAGS  # TO-DO replace this with label map  def class\_text\_to\_int(row\_label):  if row\_label == 'car':  return 1  elif row\_label == 'number plate':  return 2  else:  None  def split(df, group):  data = namedtuple('data', ['filename', 'object'])  gb = df.groupby(group)  return [data(filename, gb.get\_group(x)) for filename, x in zip(gb.groups.keys(), gb.groups)]  def create\_tf\_example(group, path):  with tf.io.gfile.GFile(os.path.join(path, '{}'.format(group.filename)), 'rb') as fid:  encoded\_jpg = fid.read()  encoded\_jpg\_io = io.BytesIO(encoded\_jpg)  image = Image.open(encoded\_jpg\_io)  width, height = image.size  filename = group.filename.encode('utf8')  image\_format = b'jpg'  xmins = []  xmaxs = []  ymins = []  ymaxs = []  classes\_text = []  classes = []  for index, row in group.object.iterrows():  xmins.append(row['xmin'] / width)  xmaxs.append(row['xmax'] / width)  ymins.append(row['ymin'] / height)  ymaxs.append(row['ymax'] / height)  classes\_text.append(row['class'].encode('utf8'))  classes.append(class\_text\_to\_int(row['class']))  tf\_example = tf.train.Example(features=tf.train.Features(feature={  'image/height': dataset\_util.int64\_feature(height),  'image/width': dataset\_util.int64\_feature(width),  'image/filename': dataset\_util.bytes\_feature(filename),  'image/source\_id': dataset\_util.bytes\_feature(filename),  'image/encoded': dataset\_util.bytes\_feature(encoded\_jpg),  'image/format': dataset\_util.bytes\_feature(image\_format),  'image/object/bbox/xmin': dataset\_util.float\_list\_feature(xmins),  'image/object/bbox/xmax': dataset\_util.float\_list\_feature(xmaxs),  'image/object/bbox/ymin': dataset\_util.float\_list\_feature(ymins),  'image/object/bbox/ymax': dataset\_util.float\_list\_feature(ymaxs),  'image/object/class/text': dataset\_util.bytes\_list\_feature(classes\_text),  'image/object/class/label': dataset\_util.int64\_list\_feature(classes),  }))  return tf\_example  def main(\_):  writer = tf.io.TFRecordWriter(FLAGS.output\_path)  path = os.path.join(os.getcwd(), FLAGS.image\_dir)  examples = pd.read\_csv(FLAGS.csv\_input)  grouped = split(examples, 'filename')  for group in grouped:  tf\_example = create\_tf\_example(group, path)  writer.write(tf\_example.SerializeToString())  writer.close()  output\_path = os.path.join(os.getcwd(), FLAGS.output\_path)  print('Successfully created the TFRecords: {}'.format(output\_path))  if \_\_name\_\_ == '\_\_main\_\_':  tf.compat.v1.app.run() |

Train and test records are created and stored in the object\_detection folder.

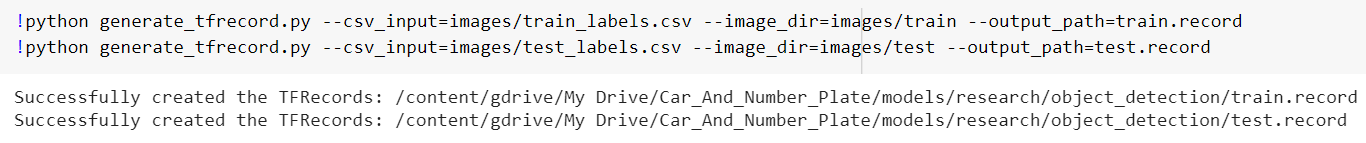


Figure 28 : Creation of TF records

**CHAPTER 5**

**TRAIN THE OBJECT DETECTION MODEL**

**5.1 OPEN TENSORBOARD:**

To background track your training checkpoints, run the code below.



Figure 29 : Load the Tensorboard

If no checkpoints present then it will not show anything until we train the model .

Once the training is completed then run the following code it will update your progressing about the training.



Figure 30: Reload the Tensorboard

**5.2 CREATE LABELMAP.PBTXT FILE:**

In /content/gdrive/My Drive/Car\_And\_Number\_Plate/models/research/object\_detection create the labelmap.pbtxt file.

As my project is about detecting Car and Vehicle Registration Plate so the content in the labelmap.pbtxt file is as follows:

labelmap.pbtxt

|  |
| --- |
| item {  id: 1  name: 'car'  }  item {  id: 2  name: 'number plate'  } |

**5.3 EDIT YOUR MODEL CONFIG FILE:**As the model used in our project is faster rcnn model so we must edit “faster\_rcnn\_inception\_v2\_pets.config” file

File after editing

|  |
| --- |
| model {  faster\_rcnn {  num\_classes: 2  image\_resizer {  keep\_aspect\_ratio\_resizer {  min\_dimension: 600  max\_dimension: 1024  }  }  feature\_extractor {  type: 'faster\_rcnn\_inception\_v2'  first\_stage\_features\_stride: 16  }  first\_stage\_anchor\_generator {  grid\_anchor\_generator {  scales: [0.25, 0.5, 1.0, 2.0]  aspect\_ratios: [0.5, 1.0, 2.0]  height\_stride: 16  width\_stride: 16  }  }  first\_stage\_box\_predictor\_conv\_hyperparams {  op: CONV  regularizer {  l2\_regularizer {  weight: 0.0  }  }  initializer {  truncated\_normal\_initializer {  stddev: 0.01  }  }  }  first\_stage\_nms\_score\_threshold: 0.0  first\_stage\_nms\_iou\_threshold: 0.7  first\_stage\_max\_proposals: 300  first\_stage\_localization\_loss\_weight: 2.0  first\_stage\_objectness\_loss\_weight: 1.0  initial\_crop\_size: 14  maxpool\_kernel\_size: 2  maxpool\_stride: 2  second\_stage\_box\_predictor {  mask\_rcnn\_box\_predictor {  use\_dropout: false  dropout\_keep\_probability: 1.0  fc\_hyperparams {  op: FC  regularizer {  l2\_regularizer {  weight: 0.0  }  }  initializer {  variance\_scaling\_initializer {  factor: 1.0  uniform: true  mode: FAN\_AVG  }  }  }  }  }  second\_stage\_post\_processing {  batch\_non\_max\_suppression {  score\_threshold: 0.0  iou\_threshold: 0.6  max\_detections\_per\_class: 100  max\_total\_detections: 300  }  score\_converter: SOFTMAX  }  second\_stage\_localization\_loss\_weight: 2.0  second\_stage\_classification\_loss\_weight: 1.0  }  }  train\_config: {  batch\_size: 1  optimizer {  momentum\_optimizer: {  learning\_rate: {  manual\_step\_learning\_rate {  initial\_learning\_rate: 0.0002  schedule {  step: 1  learning\_rate: .0002  }  schedule {  step: 900000  learning\_rate: .00002  }  schedule {  step: 1200000  learning\_rate: .000002  }  }  }  momentum\_optimizer\_value: 0.9  }  use\_moving\_average: false  }  gradient\_clipping\_by\_norm: 10.0  fine\_tune\_checkpoint: "faster\_rcnn\_inception\_v2\_coco\_2018\_01\_28/model.ckpt"  from\_detection\_checkpoint: true  # Note: The below line limits the training process to 200K steps, which we  # empirically found to be sufficient enough to train the pets dataset. This  # effectively bypasses the learning rate schedule (the learning rate will  # never decay). Remove the below line to train indefinitely.  num\_steps: 200000  data\_augmentation\_options {  random\_horizontal\_flip {  }  }  }  train\_input\_reader: {  tf\_record\_input\_reader {  input\_path: "train.record"  }  label\_map\_path: "training/labelmap.pbtxt"  }  eval\_config: {  num\_examples: 67  # Note: The below line limits the evaluation process to 10 evaluations.  # Remove the below line to evaluate indefinitely.  max\_evals: 10  }  eval\_input\_reader: {  tf\_record\_input\_reader {  input\_path: "test.record"  }  label\_map\_path: "training/labelmap.pbtxt"  shuffle: false  num\_readers: 1  } |

Major Changes to be made for the given parameters are as follows:

1. num\_classes in line 9
2. In train\_input\_reader change the input\_path(path for train.records) and label\_map\_path in line 126 and line 128 respectively.
3. In eval\_input\_reader change the input\_path(path for test.records) and label\_map\_path in line 140 and 142 respectively.
4. fine\_tune\_checkpoint in line 110.

**5.4 TRAIN YOUR MODEL:** Once all the above steps are done then you are ready to run the model finally.

For running your object detection model run the following code

We will be using train.py to train our model, but before running move the train.py from object\_detection/legacy to object\_detection directory.

Run the following code to start your training:

|  |
| --- |
| !python train.py --logtostderr --train\_dir=training/ --pipeline\_config\_path = training / faster\_rcnn\_inception\_v2\_pets.config |

Training will take several hours. Keep training the model until you get the loss less than 0.05 and steps more than 50,000. For every 5 mins the checkpoints will be stored in the training/ directory.

For this project I trained my model for 5 hours , I interrupted the execution as I got a loss value less than 0.05 and steps more than 1,00,000.

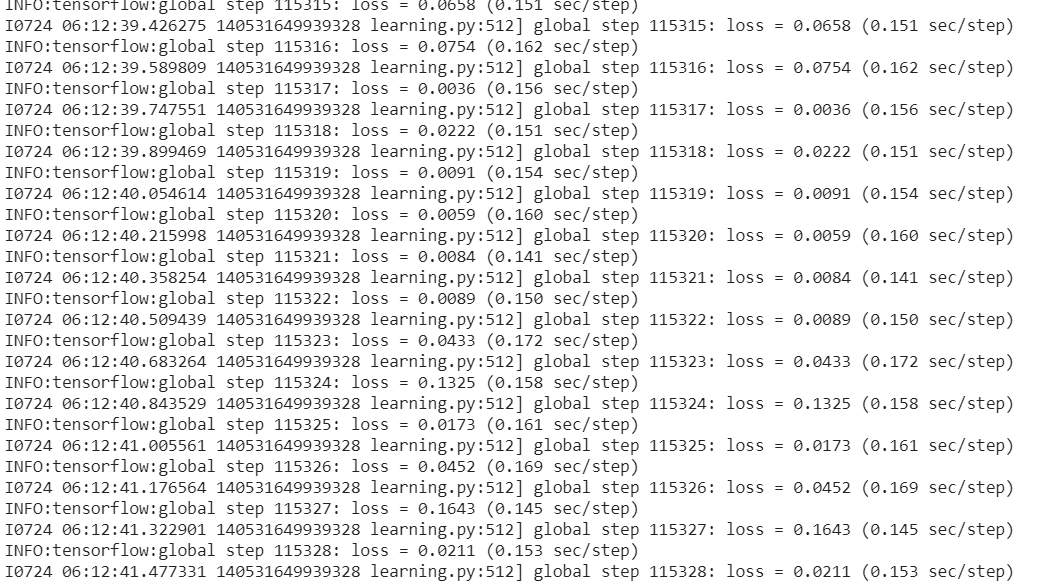


Figure 31 : Output after training

As observed above the training took 1,15,328 steps after I interrupted the execution and got a loss of 0.0211 finally.

**Note:** In case you want to continue the training where you have stopped you can execute the above code again. It will continue from the latest checkpoint that is present in the training/ directory.

In order to track the training reload the tensorboard as it was already opened

Run the cell as shown in Figure 30(page no. 30)

**After reloading the tensorboard the following observation were observed:**

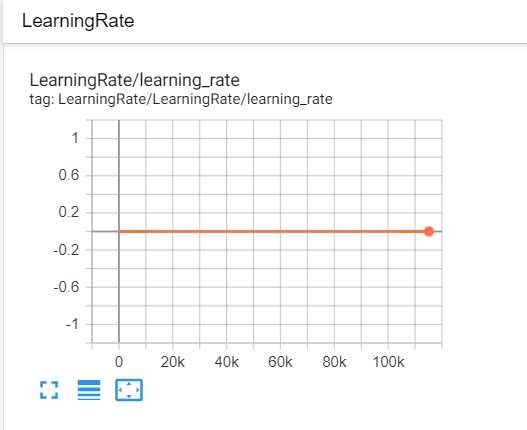


Figure 32 : Graph of Learning Rate

As the learning rate is fixed to 0.0002 in the config file so we observe the constant learning rate for each step.

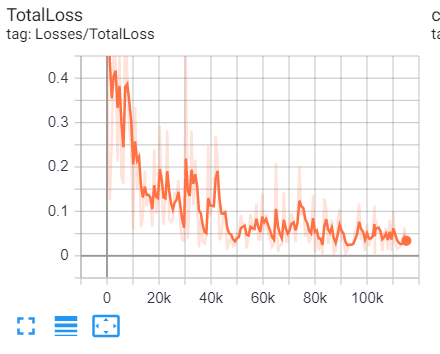


Figure 33 : Graph of Total Loss

From the graph we observe that when the no of steps increases the total loss will be decrease. But between 20K to 50K we observe a large fluctuation. Finally the Total loss is reduced to 0.03391.

**CHAPTER 6**

**TEST YOUR MODEL**

After the whole training part is over we must test the model by giving our own input images to check whether the required objects are getting detected or not.

**6.1 EXPORT THE TRAIN MODEL:**

We will export the model from training/ directory. The based on the last model checkpoint present in the training/ directory we will be exporting the model into the “frozen\_interface\_graph.pb” .

Using export\_interface\_graph.py we will export the trained model

“frozen\_interface\_graph.pb will be used for testing the model.

Check the last model checkpoint in the training/ directory

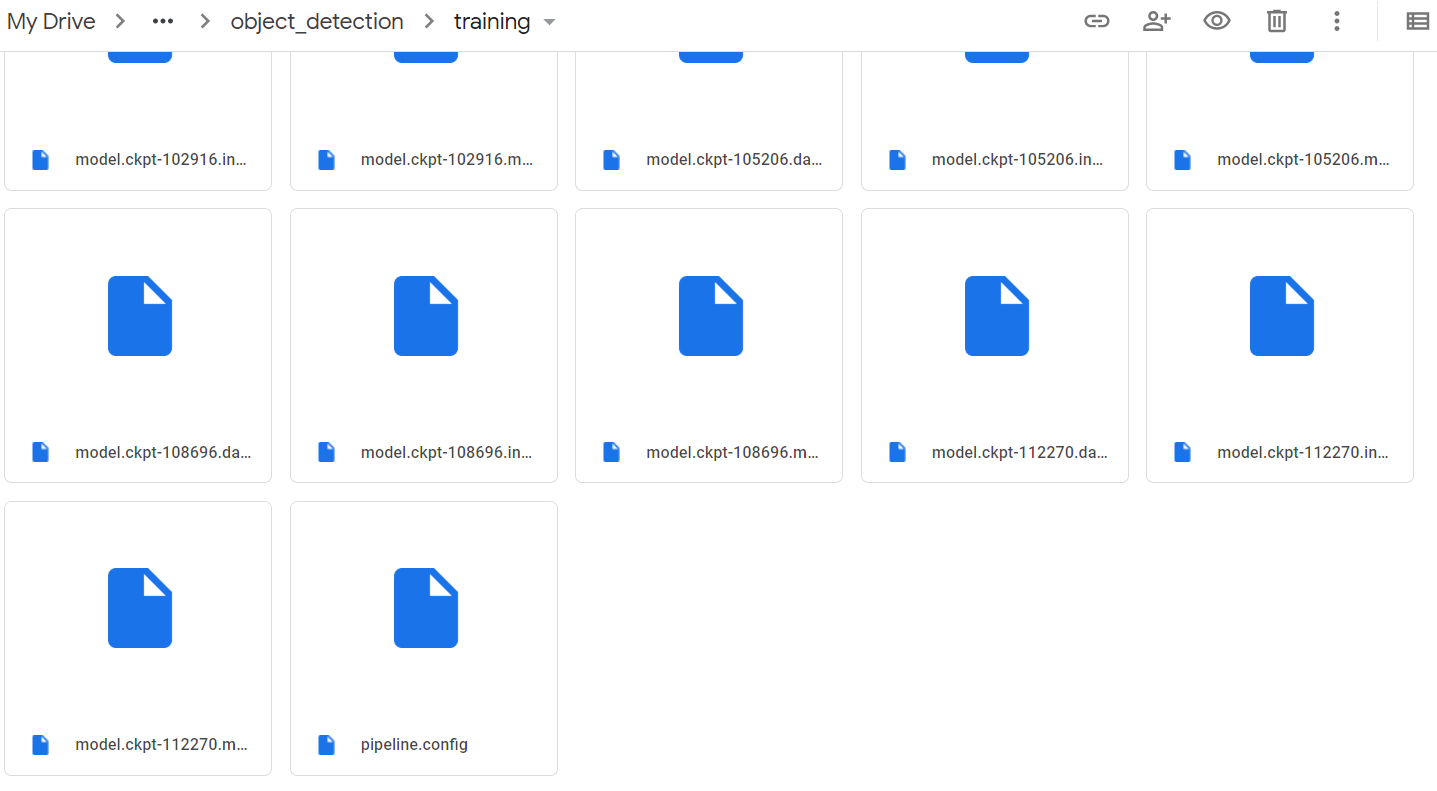


Figure 34 : Checkpoints in training directory

It is observed that the last model checkpoint step value is 112270.So using it we will export the model into “frozen\_inference\_graph.pb”

|  |
| --- |
| !python export\_inference\_graph.py --input\_type image\_tensor --pipeline\_config\_path training/faster\_rcnn\_inception\_v2\_pets.config --trained\_checkpoint\_prefix training/model.ckpt-112270 --output\_directory inference\_graph |

“frozen\_inference\_graph.pb” is stored in “inference\_graph/” directory

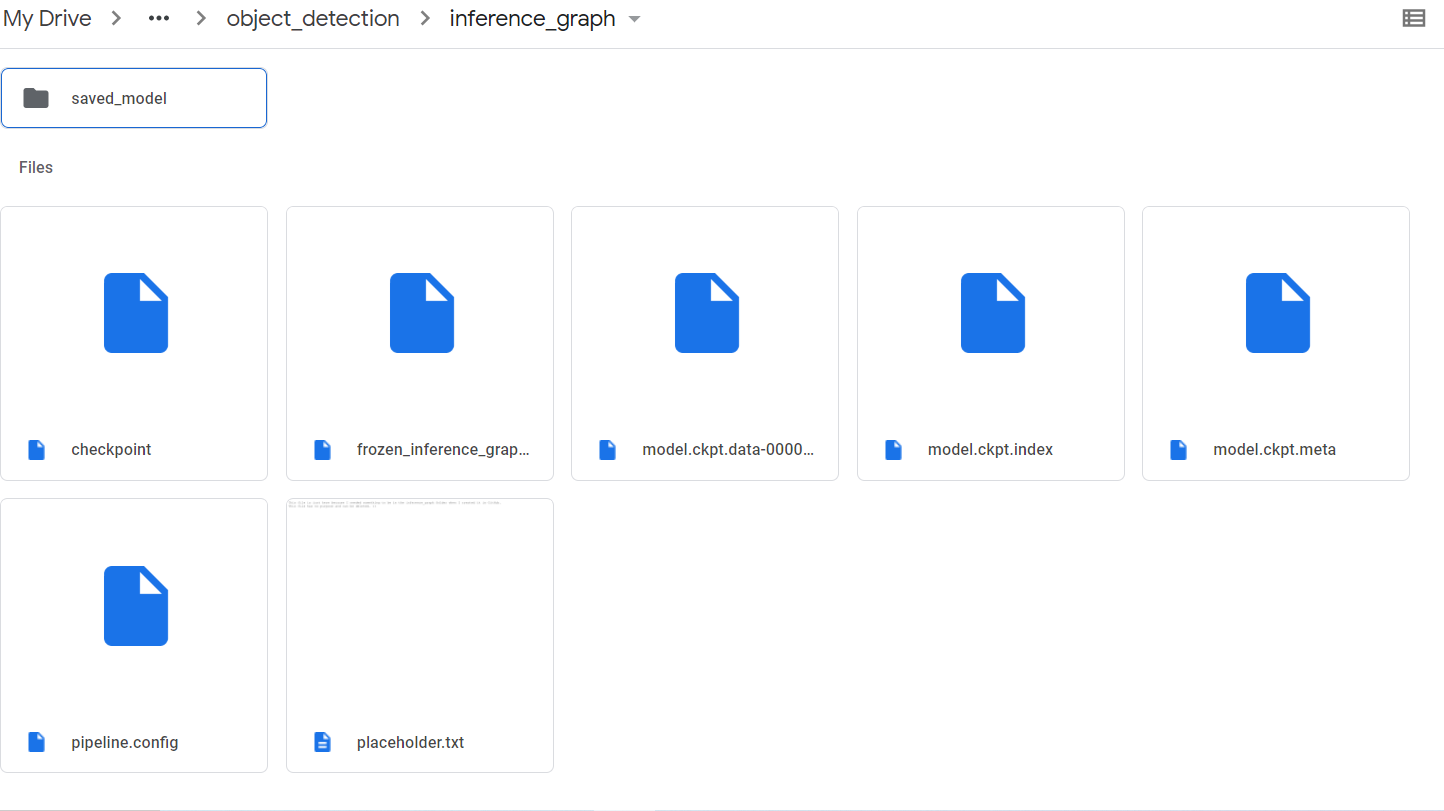
****

Figure 35 : frozen\_inference\_graph.pb in inference\_graph directory

**6.2 RUN THE OBJECT DETECTION MODEL:**

After exporting the graph into frozen\_inference\_graph.pb ,we will be using it for detecting the objects required in our own input images.

By running the following code we will get the detected objects marked in the images as the output

|  |
| --- |
| ######## Image Object Detection Using Tensorflow-trained Classifier #########  #  # Author: Evan Juras  # Date: 1/15/18  # Description:  # This program uses a TensorFlow-trained neural network to perform object detection.  # It loads the classifier and uses it to perform object detection on an image.  # It draws boxes, scores, and labels around the objects of interest in the image.  ## Some of the code is copied from Google's example at  ## https://github.com/tensorflow/models/blob/master/research/object\_detection/object\_detection\_tutorial.ipynb  ## and some is copied from Dat Tran's example at  ## https://github.com/datitran/object\_detector\_app/blob/master/object\_detection\_app.py  ## but I changed it to make it more understandable to me.  # Import packages  import os  import cv2  from google.colab.patches import cv2\_imshow  import numpy as np  import tensorflow as tf  import sys  # This is needed since the notebook is stored in the object\_detection folder.  sys.path.append("..")  # Import utilites  from utils import label\_map\_util  from utils import visualization\_utils as vis\_util  # Name of the directory containing the object detection module we're using  MODEL\_NAME = 'inference\_graph'  IMAGE\_NAME = 'image\_name.jpg'  # Grab path to current working directory  CWD\_PATH = os.getcwd()  # Path to frozen detection graph .pb file, which contains the model that is used  # for object detection.  PATH\_TO\_CKPT = os.path.join(CWD\_PATH,MODEL\_NAME,'frozen\_inference\_graph.pb')  # Path to label map file  PATH\_TO\_LABELS = os.path.join(CWD\_PATH,'training','labelmap.pbtxt')  # Path to image  PATH\_TO\_IMAGE = os.path.join(CWD\_PATH,IMAGE\_NAME)  # Number of classes the object detector can identify  NUM\_CLASSES = 2  # Load the label map.  # Label maps map indices to category names, so that when our convolution  # network predicts `5`, we know that this corresponds to `king`.  # Here we use internal utility functions, but anything that returns a  # dictionary mapping integers to appropriate string labels would be fine  label\_map = label\_map\_util.load\_labelmap(PATH\_TO\_LABELS)  categories = label\_map\_util.convert\_label\_map\_to\_categories(label\_map, max\_num\_classes=NUM\_CLASSES, use\_display\_name=True)  category\_index = label\_map\_util.create\_category\_index(categories)  # Load the Tensorflow model into memory.  detection\_graph = tf.Graph()  with detection\_graph.as\_default():  od\_graph\_def = tf.GraphDef()  with tf.gfile.GFile(PATH\_TO\_CKPT, 'rb') as fid:  serialized\_graph = fid.read()  od\_graph\_def.ParseFromString(serialized\_graph)  tf.import\_graph\_def(od\_graph\_def, name='')  sess = tf.Session(graph=detection\_graph)  # Define input and output tensors (i.e. data) for the object detection classifier  # Input tensor is the image  image\_tensor = detection\_graph.get\_tensor\_by\_name('image\_tensor:0')  # Output tensors are the detection boxes, scores, and classes  # Each box represents a part of the image where a particular object was detected  detection\_boxes = detection\_graph.get\_tensor\_by\_name('detection\_boxes:0')  # Each score represents level of confidence for each of the objects.  # The score is shown on the result image, together with the class label.  detection\_scores = detection\_graph.get\_tensor\_by\_name('detection\_scores:0')  detection\_classes = detection\_graph.get\_tensor\_by\_name('detection\_classes:0')  # Number of objects detected  num\_detections = detection\_graph.get\_tensor\_by\_name('num\_detections:0')  # Load image using OpenCV and  # expand image dimensions to have shape: [1, None, None, 3]  # i.e. a single-column array, where each item in the column has the pixel RGB value  image = cv2.imread(PATH\_TO\_IMAGE)  print(image)  image\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)  image\_expanded = np.expand\_dims(image\_rgb, axis=0)  # Perform the actual detection by running the model with the image as input  (boxes, scores, classes, num) = sess.run(  [detection\_boxes, detection\_scores, detection\_classes, num\_detections],  feed\_dict={image\_tensor: image\_expanded})  # Draw the results of the detection (aka 'visulaize the results')  vis\_util.visualize\_boxes\_and\_labels\_on\_image\_array(  image,  np.squeeze(boxes),  np.squeeze(classes).astype(np.int32),  np.squeeze(scores),  category\_index,  use\_normalized\_coordinates=True,  line\_thickness=1,  min\_score\_thresh=0.60)  # All the results have been drawn on image. Now display the image.  cv2\_imshow(image)  # Press any key to close the image  cv2.waitKey(0)  # Clean up  cv2.destroyAllWindows() |

With the “frozen\_inferance\_graph.pb” we get the detections which will help in creating the bounding boxes. “labelmap.pbtxt” helps in naming the object detected above the bounding

**Outputs:**



Figure 36: Output(1)



Figure 37: Output(2)

**CHAPTER 7**

**OBJECT DETECTION**

Object Detection is a technique associated with computer vision and image processing that performs the task of detecting instances of certain objects such as a human, vehicle, banner, building from a digital image or a video. Object detection combined with other advanced technology integrations allows us to perform face detection or pedestrian detection, popularly known as person tracking from a video. Object detection is being used in a plethora of areas such as security, human resource, healthcare, marketing, logistics and so on.

Algorithms used in object detection are

1. **Faster R-CNN**

Towards Real-Time Object Detection with Region Proposal Networks

State-of-the-art object detection networks depend on region proposal algorithms to hypothesize object locations...

This paper proposes a training mechanism that alternates fine-tuning for regional proposal tasks and fine-tuning for object detection.

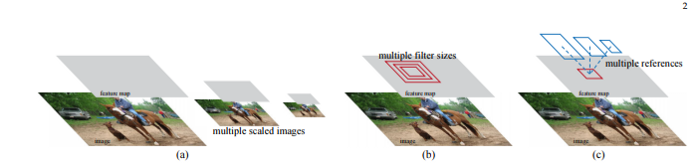


Figure 38 : Faster R-CNN(1)

The Faster R-CNN model is comprised of two modules: a deep convolutional network responsible for proposing the regions, and a Fast R-CNN detector that uses the regions. The Region Proposal Network takes an image as input and generates an output of rectangular object proposals. Each of the rectangles has an objectness score.

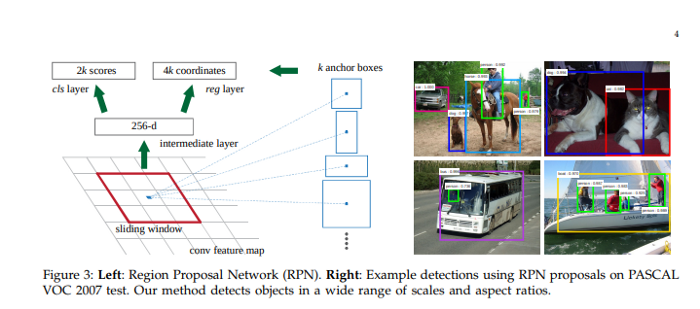


Figure 39 : Faster R-CNN(2)

1. **SSD: Single Shot MultiBox Detector**

We present a method for detecting objects in images using a single deep neural network. Our approach, named SSD...

This paper presents a model to predict objects in images using a single deep neural network. The network generates scores for the presence of each object category using small convolutional filters applied to feature maps.

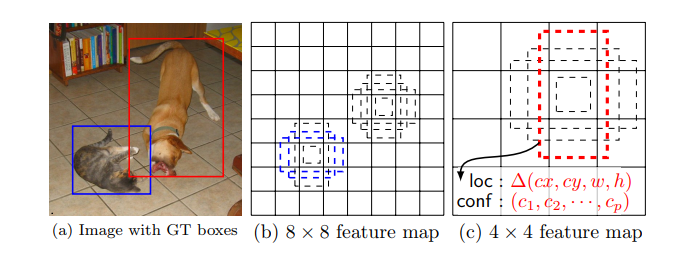


Figure 40: SSD(1)

This approach uses a feed-forward convolutional neural network that produces a collection of bounding boxes and scores for the presence of certain objects. Convolutional feature layers are added to allow for feature detection at multiple scales. In this model, each feature map cell is linked to a set of default bounding boxes.

1. **You Only Look Once(YOLO)**

This paper proposes a single neural network to predict bounding boxes and class probabilities from an image in a single evaluation.

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform...

The YOLO models process 45 frames per second in real-time. YOLO views image detection as a regression problem, which makes its pipeline quite simple. It’s extremely fast because of this simple pipeline.

It can process a streaming video in real-time with a latency of less than 25 seconds. During the training process, YOLO sees the entire image and is, therefore, able to include the context in object detection.

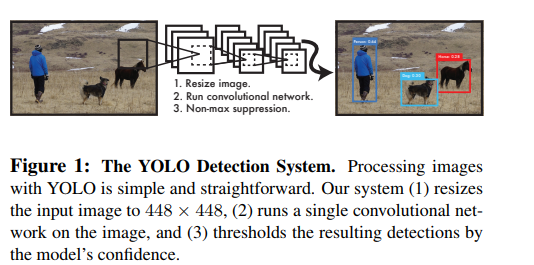


Figure 41: YOLO(1)

In YOLO, each bounding box is predicted by features from the entire image. Each bounding box has 5 predictions; x, y, w, h, and confidence. (x, y) represents the center of the bounding box relative to the bounds of the grid cell. W and h are the predicted width and height of the whole image.

This model is implemented as a convolutional neural network and evaluated on the PASCAL VOC detection dataset. The convolutional layers of the network are responsible for extracting the features, while the fully connected layers predict the coordinates and output probabilities.

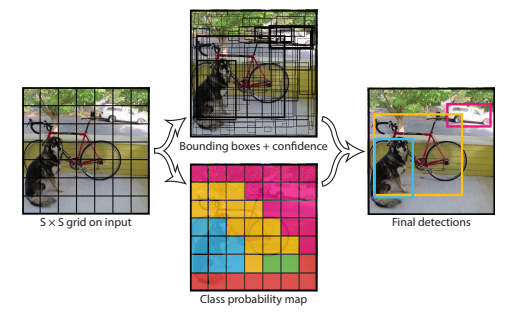


Figure 42:YOLO(2)

**Confidence score**

To interpret these results, we can look at the score and the location for each detected object. The score is a number between 0 and 1 that indicates confidence that the object was genuinely detected. In percentage it is 0% to 100%.

The Confidence Score tells how accurate the object is detected. As observed above in figure 36 and figure 37, We can see the percentage of the bounding boxes which is nothing but the Confidence Score. But only the region of detection with confidence score above 60% has a bounding box.

**CONCLUSION:**

It is concluded that the Object detection model of detecting the Car and Vehicle Registration plate has been done successful. The training of the model has been done in more than 1,00,000 steps and got a loss less than 0.05 . This Model is helpful for the traffic police in order to monitor the traffic well. Traffic Police will be able to find out the owner of the vehicle who is not following the rules in the traffic signal through Vehicle registration Plate details. Faster R-CNN Model was used because we will get more accurate results compared with other models like SSD and YOLO.

**References:**

<https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10>

<https://storage.googleapis.com/openimages/web/visualizer/index.html?set=train&type=segmentation&r=false&c=%2Fm%2F01m4t>