

**Summer Project On**  
**Automatic Number Plate Detection**  
**By**  
**Ganesh Chaudhari (2021510007)**  
**Tirsha Das (2021510008)**

Under the guidance of  
**Internal Supervisor**

**Prof.Pallavi Thakur**



Department of Master Of Computer Application  
Sardar Patel Institute of Technology  
Autonomous Institute Affiliated to Mumbai University  
2022-23

## **CERTIFICATE OF APPROVAL**

This is to certify that the following students

**Ganesh Chaudhari (2021510007)**  
**Tirsha Das (2021510008)**

Have satisfactorily carried out work on the project  
entitled

### **“Automatic Number Plate Detection”**

Towards the fulfilment of project, as laid down  
by

Sardar Patel Institute of Technology  
during year  
2022-23.

Project Guide:  
Prof.Pallavi Thakur

## **PROJECT APPROVAL CERTIFICATE**

This is to certify that the following students

**Ganesh Chaudhari (2021510007)**  
**Tirsha Das (2021510008)**

Have successfully completed the Project report on

**“Automatic Number Plate Detection”,**

which is found to be satisfactory and is approved

at

SARDAR PATEL INSTITUTE OF TECHNOLOGY,  
ANDHERI (W), MUMBAI

INTERNAL EXAMINER

EXTERNAL EXAMINER

HEAD OF DEPARTMENT

PRINCIPAL

# Contents

<b>Abstract</b>	i
<b>Objectives</b>	i
<b>List Of Figures</b>	ii
<b>List Of Tables</b>	ii
<b>1 Introduction</b>	1
1.1 Problem Definition . . . . .	1
1.2 Objectives and Scope . . . . .	1
1.2.1 Objectives . . . . .	1
1.2.2 Scope . . . . .	1
1.3 Existing System . . . . .	2
1.4 Proposed System . . . . .	2
1.5 System Requirements . . . . .	3
<b>2 Software Requirement Specification (SRS) and Design</b>	4
2.1 Purpose . . . . .	4
2.2 Definition . . . . .	4
2.3 Overall Description . . . . .	4
2.3.1 Product Functions . . . . .	4
2.3.2 User Characteristics . . . . .	5
<b>3 Project Analysis and Design</b>	6
3.1 Methodologies Adapted . . . . .	6
3.2 Modules . . . . .	7
3.2.1 Activity diagram . . . . .	7
3.2.2 Work Breakdown Structure . . . . .	8
3.2.3 PERT Chart . . . . .	9
3.2.4 Gantt Chart . . . . .	9
3.2.5 Use-Case . . . . .	10
<b>4 Project Implementation and Testing</b>	13
4.1 License Plate Detection . . . . .	13
4.2 License Plate Region . . . . .	14
4.3 Number Plate Detected . . . . .	15
4.4 Marks . . . . .	16
4.5 Code 1 . . . . .	17
4.6 Code 2 . . . . .	17
4.7 Code 3 . . . . .	18
4.8 Code 4 . . . . .	18
4.9 Code 5 . . . . .	19
4.10 Code 6 . . . . .	19
4.11 Code 7 . . . . .	20

4.12	Code 8 . . . . .	20
4.13	Code 9 . . . . .	21
4.14	Code 10 . . . . .	21
4.15	Code 11 . . . . .	22
4.16	Code 12 . . . . .	22
4.17	Code 13 . . . . .	23
4.18	Code 14 . . . . .	23
4.19	Code 15 . . . . .	24
4.20	Code 16 . . . . .	24
4.21	Code 17 . . . . .	25
4.22	Code 18 . . . . .	25
4.23	Code 19 . . . . .	26
4.24	Code 20 . . . . .	26
<b>5</b>	<b>Test Cases</b>	<b>27</b>
<b>6</b>	<b>Limitations</b>	<b>28</b>
<b>7</b>	<b>Future Enhancements</b>	<b>28</b>
<b>8</b>	<b>User Manual</b>	<b>29</b>
<b>9</b>	<b>Bibliography</b>	<b>30</b>
9.1	Web References . . . . .	30

## **Abstract**

Automatic Number Plate Recognition (ANPR) is an image processing technology which uses number (license) plate to identify the vehicle. The main purpose is to design an efficient automatic authorized vehicle identification system by using the vehicle number plate.

The system is implemented on the entrance for security control of a highly restricted area like military zones or area around top government offices e.g. Parliament, Supreme Court etc. The developed system first detects the license plate and captures them. The images are stored in folder and the number of license plate are store in excel file

Vehicle number plate region is extracted using the image segmentation in an image. Optical character recognition technique is used for the character recognition. It is observed from the experiment that the developed system successfully detects and recognize the vehicle number plate on real images.

## **Objectives**

The Automatic Number Plate Detection is used

- To provide help to detect number on license plate.
- To provide keep a track of all the vehicles passing
- To provide images of all the license plate captured.
- To provide an excel sheet of with number of license plate

## List of Figures

3.1.1Diagrammatic Representation of Waterfall Model . . . . .	6
3.2.1Activity Diagram . . . . .	7
3.2.2Work Breakdown Structure . . . . .	8
3.2.3PERT Chart . . . . .	9
3.2.4Gantt Chart . . . . .	9
3.2.5Use-Case Diagram . . . . .	10
4.1.1 License Plate Detection . . . . .	13
4.2.1 License Plate Region . . . . .	14
4.3.1Number Plate Detected . . . . .	15
4.4.1Marks View . . . . .	16

## List of Tables

1.5.1 Hardware Requirements . . . . .	3
4.2.1 Use Case Table - Scan Image . . . . .	11
4.2.2 Use Case Table - Detect Number Plate . . . . .	11
4.2.3 Use Case Table - Send Image . . . . .	11
4.2.4 Use Case Table - Convert Image To text . . . . .	12
4.2.5 Use Case Table - Save Text Of Number Plate . . . . .	12
6.1 Test Case . . . . .	27

## 1 Introduction

### 1.1 Problem Definition

To help keep a track of the all vehicles license plate in any location with help of appropriate camera. It will be able to detect number of license plate from a video as well from picture.

### 1.2 Objectives and Scope

#### 1.2.1 Objectives

The Automatic Number Plate Detection

- To provide security staff an upper hand in tracking the vehicles in parking lots, restricted zone, etc.
- To have list of all the vehicles number plate along with photos
- To provide help in detecting suspicious patterns in visiting highway parking can be related to criminal activities.
- To provide to municipalities to keep track of incoming and outgoing traffic.

#### 1.2.2 Scope

The user will have access to all the record and images and excel sheet and can mail to any person according to the need.

With help of the project keeping a track of all the vehicles becomes easy.

### 1.3 Existing System

Currently there are few automatic number plate detection available in market, but the lack some or the other feature.

Some of the disadvantages of existing system are as follows :

- Not Organised  
The number's of license plate becomes difficult to keep a track of, as there a lot of data involved .
- Non Sharing  
In available number plate detection there is no way to send email of records to anyone.
- No Image Records  
It is seen sometime the automatic number plate detection fails to capture the number and there is no way to cross check it.

### 1.4 Proposed System

Automatic Number Plate Recognition system i.e. ANPR system is an image processing technology. In this technology we use license plate of vehicle to recognize the vehicle. Automatic Number Plate Recognition system is used in various areas nowadays such as automatic toll collection, Border crossings, parking system, Traffic control, stolen cars tracking, maintaining traffic activities and law enforcement etc. several techniques have been proposed for number plate recognition.

Some of the advantages of our system are as follows :

- The user will no get all the images of number plates, so even if there is fault in detection of number, or the number is not capture, the user can use the images to keep a track of things.
- Users can get mail of all the license plate detected in a excel sheet.

## 1.5 System Requirements

- Hardware Requirements

Table 1.5.1: Hardware Requirements

Processor	Dual Core Processor or Above
RAM	Minimum 4 GB RAM
Storage	Minimum 10 GB Hard Disk Space for smooth run
Camera	As good as possible

## **2 Software Requirement Specification (SRS) and Design**

### **2.1 Purpose**

In this project, we are going to create an Automatic Number Plate Recognition which will detect the number plate and will display the characters of the number plate on the screen. This extraction of characters is done using Optical Character Recognition (OCR). The number plate detection will be done through some algorithm/technique with the help of Python language .

### **2.2 Definition**

To build a automatic number plate detection for making the task easy.

### **2.3 Overall Description**

#### **2.3.1 Product Functions**

The product function includes:

1. Number Plate Detection from image: The system will detect number plate from an image.
2. Number Plate Detection from video: The system will detect number plate from an video.
3. Number Plate Detection live: The system will detect number plate at run time.
4. Image Storage: All the license plate captured are stored in a folder.
5. Excel Storage: All the number captured from license plate captured are stored in excel sheet.

### **2.3.2 User Characteristics**

There can be any types of users:

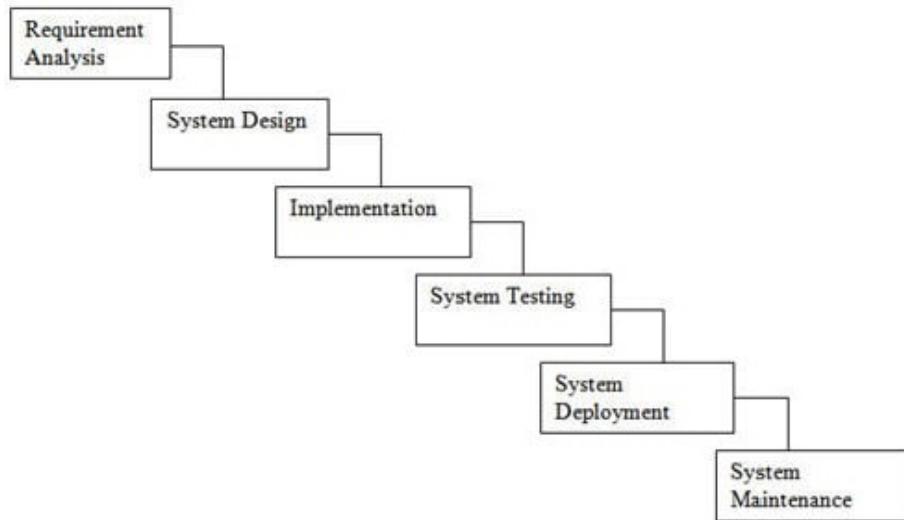
- The user using the system will have all access to the data.

### 3 Project Analysis and Design

#### 3.1 Methodologies Adapted

In Waterfall model, very less customer interaction is involved during the development of the product. Once the product is ready then only it can be demonstrated to the end users.

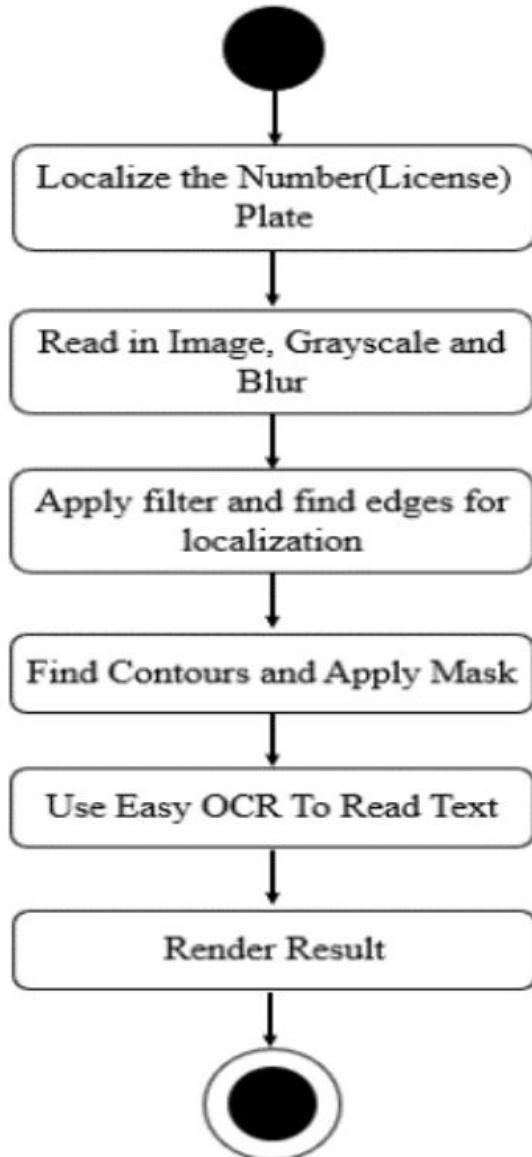
Once the product is developed and if any failure occurs then the cost of such issues is very high, because we need to update everything from document till the logic.



3.1.1: Diagrammatic Representation of Waterfall Model

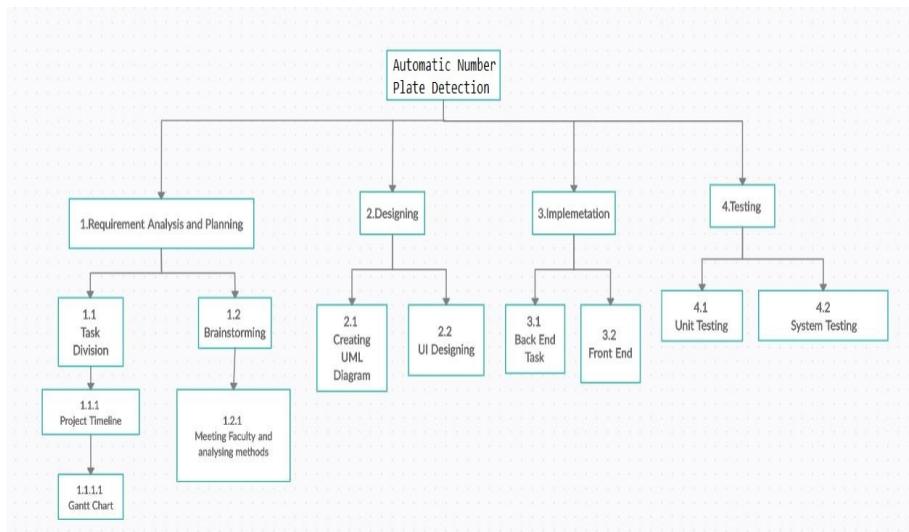
### 3.2 Modules

#### 3.2.1 Activity diagram



3.2.1: Activity Diagram

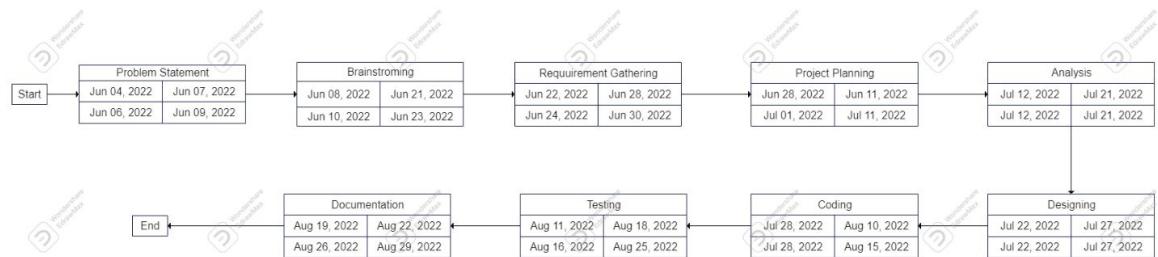
### 3.2.2 Work Breakdown Structure



3.2.2: Work Breakdown Structure

## Automatic Number Plate Detection

### 3.2.3 PERT Chart



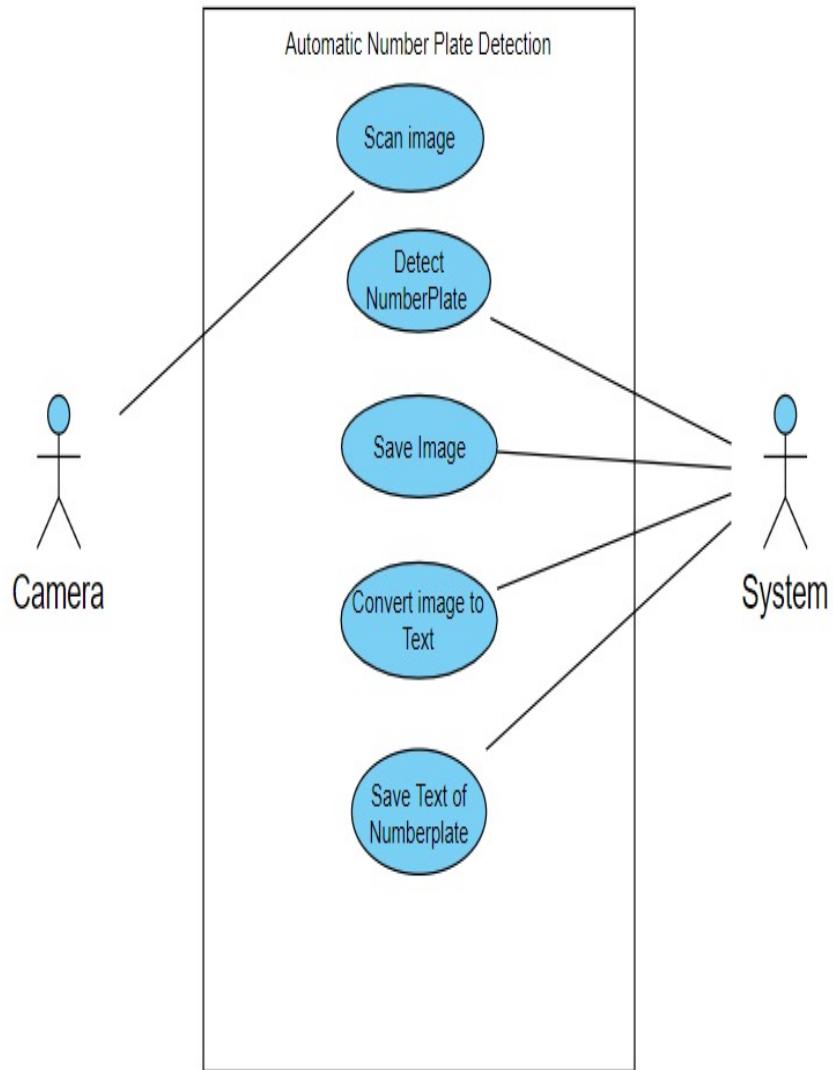
3.2.3: PERT Chart

### 3.2.4 Gantt Chart



3.2.4: Gantt Chart

### 3.2.5 Use-Case



3.2.5: Use-Case Diagram

## Automatic Number Plate Detection

---

Use Cases:

1. Scan Image
2. Detect Number Plate
3. Send Image
4. Convert Image To text
5. Save Text Of Number Plate

Table 4.2.1: Use Case Table - Scan Image

Use Case ID	1
Use Case Name	Scan Image
Actor	Camera
Pre-Condition	Camera should be in Connected
Post-Condition	Camera Should be workingn

Table 4.2.2: Use Case Table - Detect Number Plate

Use Case ID	2
Use Case Name	Detect Number Plate
Actor	System
Pre-Condition	They must register themselves first
Post-Condition	User can view its details and marks and view and upload Companies
Flow of events	Login,Register or Edit details and marks or upload and view companies

Table 4.2.3: Use Case Table - Send Image

Use Case ID	3
Use Case Name	Send Image
Actor	TPC and non-TPC
Pre-Condition	Login
Post-Condition	User can view or edit the profile

Table 4.2.4: Use Case Table - Convert Image To text

Use Case ID	4
Use Case Name	Convert Image To text
Actor	TPC and non-TPC
Pre-Condition	Login
Post-Condition	User can view and edit the marks

Table 4.2.5: Use Case Table - Save Text Of Number Plate

Use Case ID	5
Use Case Name	Save Text Of Number Plate
Actor	TPC
Pre-Condition	Login
Post-Condition	Can view marks

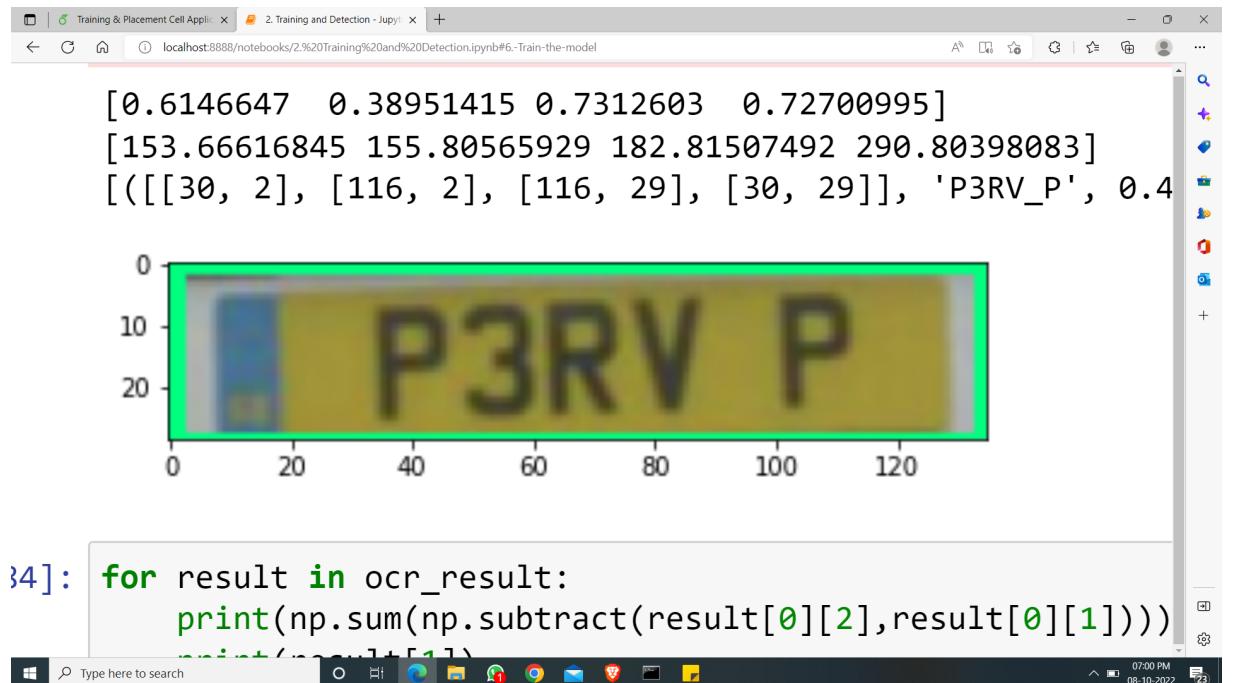
## 4 Project Implementation and Testing

### 4.1 License Plate Detection



4.1.1: License Plate Detection

## 4.2 License Plate Region



The screenshot shows a Jupyter Notebook cell with the following output:

```
[0.6146647 0.38951415 0.7312603 0.72700995]
[153.66616845 155.80565929 182.81507492 290.80398083]
[[[30, 2], [116, 2], [116, 29], [30, 29]], 'P3RV_P', 0.4]
```

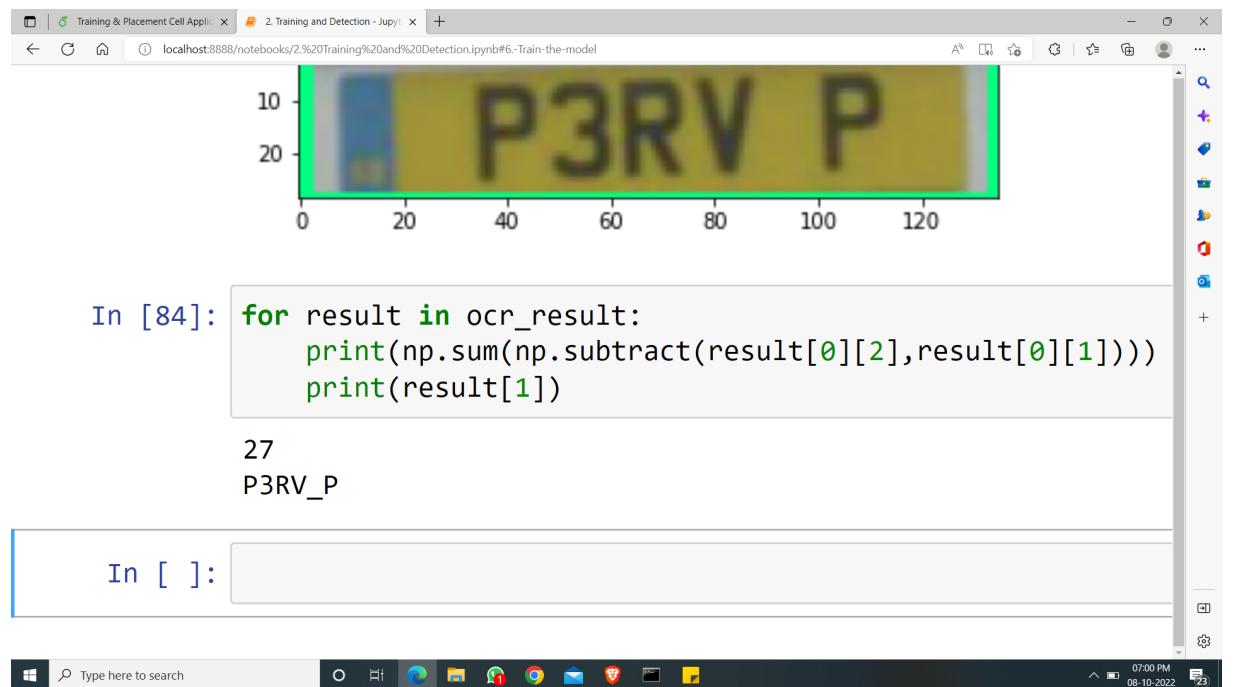
Below the text output is a visualization of a license plate. The image is a yellow rectangle with a green border. Inside the image, the letters "P3RV\_P" are printed in black. The image has coordinate axes ranging from 0 to 120 on both the x and y axes.

```
34]: for result in ocr_result:
    print(np.sum(np.subtract(result[0][2], result[0][1])))
```

The notebook interface includes a search bar at the bottom left, a taskbar with various icons, and a status bar at the bottom right indicating the date and time.

4.2.1: License Plate Region

### 4.3 Number Plate Detected



```
In [84]: for result in ocr_result:  
    print(np.sum(np.subtract(result[0][2],result[0][1])))  
    print(result[1])  
  
27  
P3RV_P
```

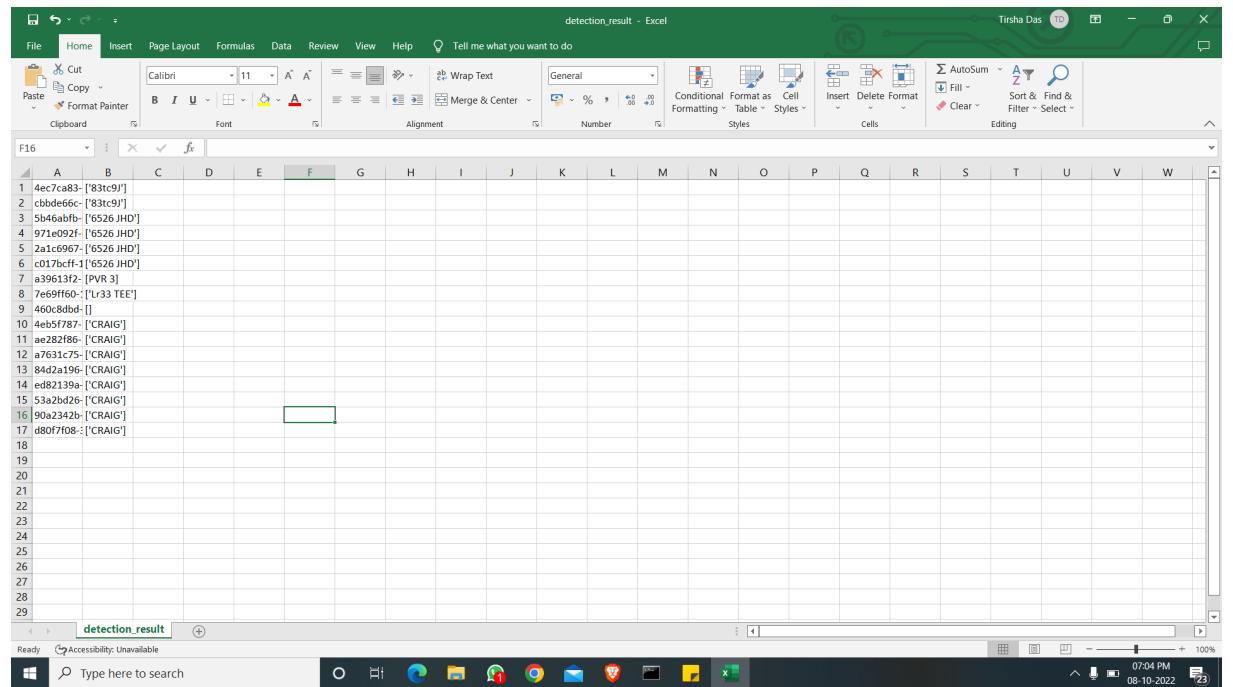
In [ ]:

4.3.1: Number Plate Detected

## Automatic Number Plate Detection

---

### 4.4 Marks



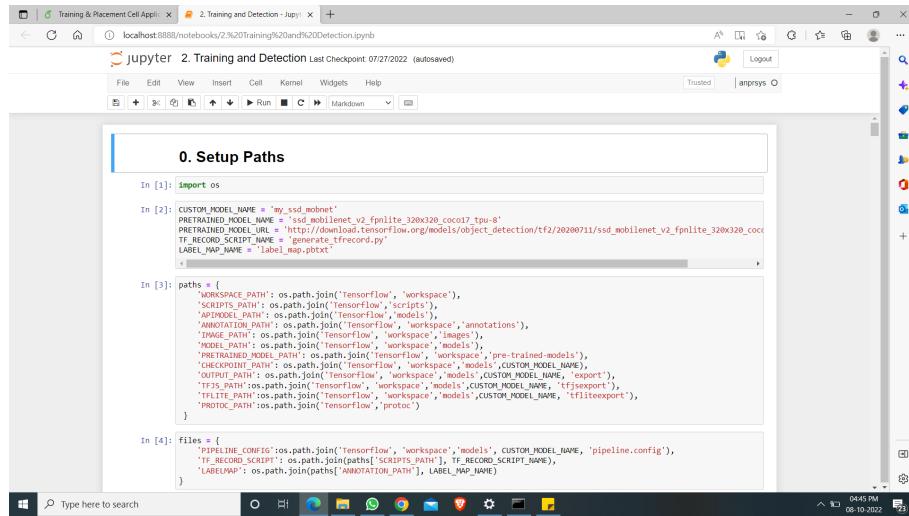
A screenshot of a Microsoft Excel spreadsheet titled "detection\_result - Excel". The spreadsheet contains a single column of data labeled "Marks" in column A. The data consists of 17 rows of JSON objects, each containing a unique identifier and a list of characters. The identifiers range from 1 to 17. The data is as follows:

	Marks
1	4ec7ca83-['83tc9!']
2	cbbde6fc-['83tc9!']
3	5b46abfb-['6526 JHD']
4	971e092f-['6526 JHD']
5	2a1c6967-['6526 JHD']
6	c017bcff-1['6526 JHD']
7	a39613f2- [PVR 3]
8	7e69f160-:[Lr33 TEE]
9	460c8dbd[]
10	4eb5f787- ['CRAIG']
11	a282f86- ['CRAIG']
12	a7631c75- ['CRAIG']
13	84d2a196- ['CRAIG']
14	e9d82139e- ['CRAIG']
15	53a2bd26- ['CRAIG']
16	90a2342b- ['CRAIG']
17	d80f7f08-:[CRAIG]

4.4.1: Marks View

## Automatic Number Plate Detection

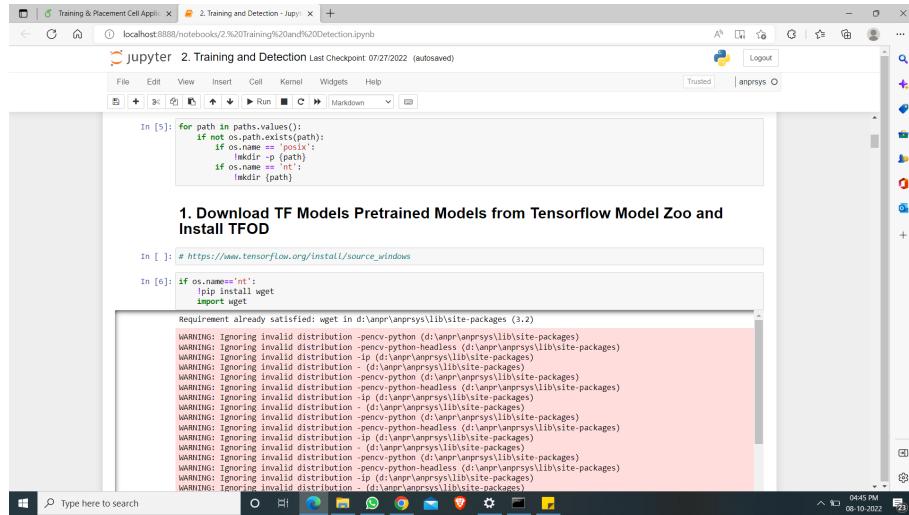
### 4.5 Code 1



The screenshot shows a Jupyter Notebook interface with two tabs: 'Training & Placement Cell App' and '2. Training and Detection - Jupyter'. The current tab is '2. Training and Detection - Jupyter'. The notebook contains the following code:

```
In [1]: import os  
  
In [2]: CUSTOM_MODEL_NAME = 'my_ssd_mobilenet'  
PRETRAINED_MODEL_NAME = 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8'  
PRETRAINED_MODEL_URL = 'http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz'  
TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'  
LABEL_MAP_NAME = 'label_map.pbtxt'  
  
In [3]: paths = {  
    'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),  
    'SCRIPTS_PATH': os.path.join('Tensorflow', 'scripts'),  
    'APIMODEL_PATH': os.path.join('Tensorflow', 'models'),  
    'ANNOTATION_PATH': os.path.join('Tensorflow', 'workspace', 'annotations'),  
    'IMAGE_PATH': os.path.join('Tensorflow', 'workspace', 'images'),  
    'PROTO_PATH': os.path.join('Tensorflow', 'protos'),  
    'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow', 'workspace', 'pre-trained-models'),  
    'CHECKPOINT_PATH': os.path.join('Tensorflow', 'workspace', 'models', CUSTOM_MODEL_NAME),  
    'CUSTOM_MODEL_NAME': 'my_ssd_mobilenet',  
    'PIPELINE_CONFIG_PATH': os.path.join(CUSTOM_MODEL_NAME, 'pipeline.config'),  
    'TF_RECORD_SCRIPT': os.path.join(paths['SCRIPTS_PATH'], TF_RECORD_SCRIPT_NAME),  
    'TF_LABELMAP': os.path.join(paths['ANNOTATION_PATH'], LABEL_MAP_NAME)  
}  
  
In [4]: files = [  
    'PIPELINE_CONFIG_PATH', os.path.join('Tensorflow', 'workspace', 'models', CUSTOM_MODEL_NAME, 'pipeline.config'),  
    'TF_RECORD_SCRIPT', os.path.join(paths['SCRIPTS_PATH'], TF_RECORD_SCRIPT_NAME),  
    'LABEL_MAP', os.path.join(paths['ANNOTATION_PATH'], LABEL_MAP_NAME)  
]
```

### 4.6 Code 2

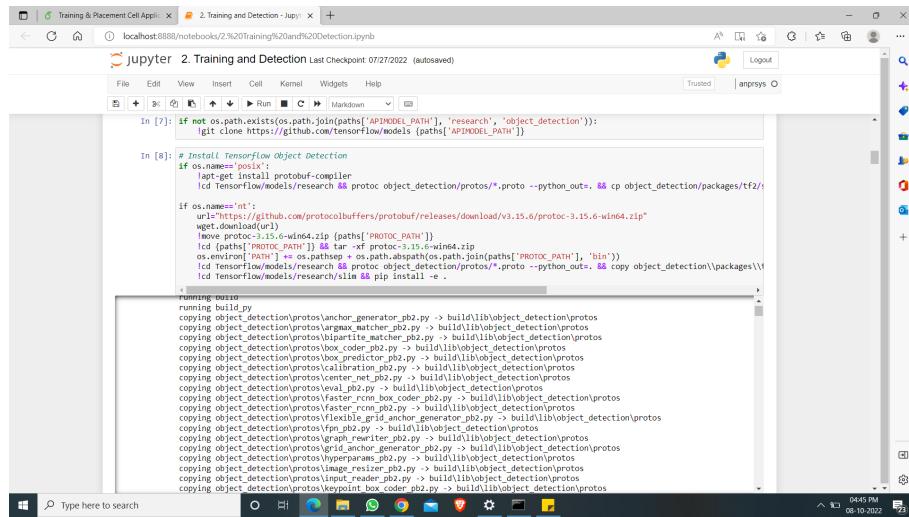


The screenshot shows a Jupyter Notebook interface with two tabs: 'Training & Placement Cell App' and '2. Training and Detection - Jupyter'. The current tab is '2. Training and Detection - Jupyter'. The notebook contains the following code:

```
In [5]: for path in paths.values():  
    if not os.path.exists(path):  
        if os.name == 'nt':  
            mkdir -p {path}  
        if os.name == 'nt':  
            mkdir {path}  
  
1. Download TF Models Pretrained Models from Tensorflow Model Zoo and  
Install TFOD  
  
In [1]: # https://www.tensorflow.org/install/source_windows  
  
In [6]: if os.name=='nt':  
    pip install wget  
    import wget  
  
Requirement already satisfied: wget in d:\anaconda\envs\tfod\lib\site-packages (3.2)  
WARNING: Ignoring invalid distribution: pency-python (d:\anaconda\envs\tfod\lib\site-packages)  
WARNING: Ignoring invalid distribution: pency-python-headless (d:\anaconda\envs\tfod\lib\site-packages)  
WARNING: Ignoring invalid distribution: -p (d:\anaconda\envs\tfod\lib\site-packages)  
WARNING: Ignoring invalid distribution: - (d:\anaconda\envs\tfod\lib\site-packages)  
WARNING: Ignoring invalid distribution: -p (d:\anaconda\envs\tfod\lib\site-packages)  
WARNING: Ignoring invalid distribution: - (d:\anaconda\envs\tfod\lib\site-packages)  
WARNING: Ignoring invalid distribution: -p (d:\anaconda\envs\tfod\lib\site-packages)  
WARNING: Ignoring invalid distribution: - (d:\anaconda\envs\tfod\lib\site-packages)  
WARNING: Ignoring invalid distribution: -p (d:\anaconda\envs\tfod\lib\site-packages)  
WARNING: Ignoring invalid distribution: - (d:\anaconda\envs\tfod\lib\site-packages)  
WARNING: Ignoring invalid distribution: -p (d:\anaconda\envs\tfod\lib\site-packages)  
WARNING: Ignoring invalid distribution: - (d:\anaconda\envs\tfod\lib\site-packages)  
WARNING: Ignoring invalid distribution: -p (d:\anaconda\envs\tfod\lib\site-packages)  
WARNING: Ignoring invalid distribution: - (d:\anaconda\envs\tfod\lib\site-packages)  
WARNING: Ignoring invalid distribution: -p (d:\anaconda\envs\tfod\lib\site-packages)  
WARNING: Ignoring invalid distribution: - (d:\anaconda\envs\tfod\lib\site-packages)
```

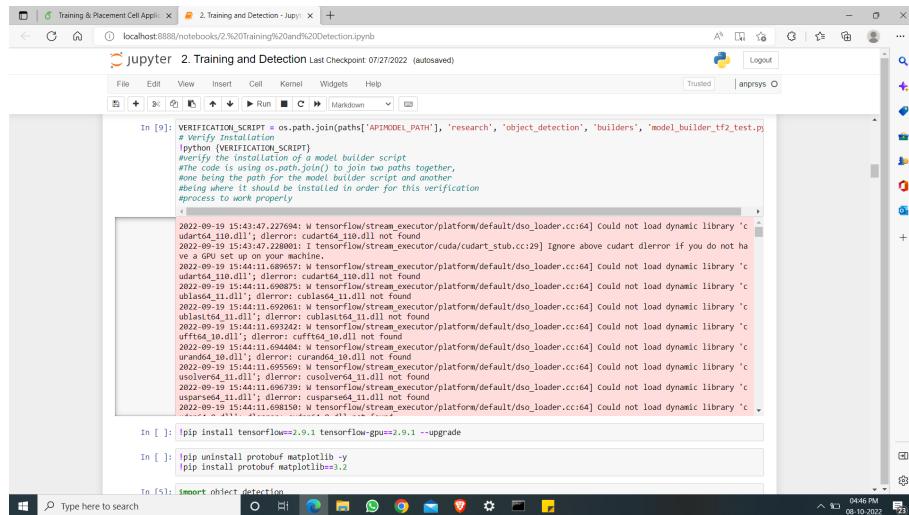
## Automatic Number Plate Detection

### 4.7 Code 3



```
In [7]: if not os.path.exists(os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection')):  
    !git clone https://github.com/tensorflow/models [paths['APIMODEL_PATH']]  
  
In [8]: # Install TensorFlow Object Detection  
# If no --name='posix':  
#     !apt-get install protobuf-compiler  
#     !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --python_out=. && cp object_detection/packages/tf2/*.  
if os.name=='nt':  
    url='https://github.com/protocolbuffers/protobuf/releases/download/v3.15.0/protobuf-3.15.0-win64.zip'  
    !wget -c -O protobuf-win64.zip [path['PROTOC_PATH']]  
    !tar -xvf protobuf-3.15.0-win64.zip  
    os.environ['PATH'] = os.pathsep + os.path.abspath(os.path.join(path['PROTOC_PATH'], 'bin'))  
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --python_out=. && copy object_detection\packages\*\*\*.  
    !cd Tensorflow\models\research\slim && pip install -e .  
  
!cd Tensorflow\models\research\slim  
!python build.py  
copying object_detection\protos\anchor_generator_pb2.py -> build\lib\object_detection\protos  
copying object_detection\protos\argmax_matcher_pb2.py -> build\lib\object_detection\protos  
copying object_detection\protos\clip_loss_pb2.py -> build\lib\object_detection\protos  
copying object_detection\protos\classification_loss_pb2.py -> build\lib\object_detection\protos  
copying object_detection\protos\box_predictor_pb2.py -> build\lib\object_detection\protos  
copying object_detection\protos\box_coder_pb2.py -> build\lib\object_detection\protos  
copying object_detection\protos\calibration_pb2.py -> build\lib\object_detection\protos  
copying object_detection\protos\eval_pb2.py -> build\lib\object_detection\protos  
copying object_detection\protos\faster_rcnn_box_code_pb2.py -> build\lib\object_detection\protos  
copying object_detection\protos\faster_rcnn_feature_extractor_pb2.py -> build\lib\object_detection\protos  
copying object_detection\protos\graph_rewriter_pb2.py -> build\lib\object_detection\protos  
copying object_detection\protos\image_resizer_pb2.py -> build\lib\object_detection\protos  
copying object_detection\protos\hyperparams_pb2.py -> build\lib\object_detection\protos  
copying object_detection\protos\input_reader_pb2.py -> build\lib\object_detection\protos  
copying object_detection\protos\keypoint_box_coder_pb2.py -> build\lib\object_detection\protos
```

### 4.8 Code 4



```
In [9]: VERIFICATION_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection', 'builders', 'model_builder_tf2_test.py')  
# Verify Initialization  
# python [VERIFICATION_SCRIPT]  
# Verify initialization of a model builder script  
# The code is using os.path.join() to join two paths together,  
# one being the path for the model builder script and another  
# stating where it should be installed in order for this verification  
# process to work properly  
  
2022-09-19 15:42:47.227694: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlopen: cudart64_110.dll not found  
2022-09-19 15:42:47.228001: W tensorflow/stream_executor/cuda/cuda_stub.cc:29] Ignore above cudart dlopen error if you do not have a CUDA driver installed on your machine  
2022-09-19 15:44:11.689057: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlopen: cudart64_110.dll not found  
2022-09-19 15:44:11.689085: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cublas64_11.dll'; dlopen: cublas64_11.dll not found  
2022-09-19 15:44:11.689103: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cufft64_10.dll'; dlopen: cufft64_10.dll not found  
2022-09-19 15:44:11.694040: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'curand64_10.dll'; dlopen: curand64_10.dll not found  
2022-09-19 15:44:11.694058: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cusolver64_11.dll'; dlopen: cusolver64_11.dll not found  
2022-09-19 15:44:11.698793: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cusparse64_11.dll'; dlopen: cusparse64_11.dll not found  
2022-09-19 15:44:11.698810: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cupti64_11.dll'; dlopen: cupti64_11.dll not found  
  
In [1]: !pip install tensorflow<2.9.1 tensorflow-gpu<2.9.1 --upgrade  
In [1]: !pip uninstall protobuf matplotlib -y  
In [1]: !pip install protobuf matplotlib==3.3  
  
In [51]: import object_detection
```

## Automatic Number Plate Detection

## 4.9 Code 5

The screenshot shows a Jupyter Notebook interface with the title "2. Training and Detection - Jupyter". The notebook has a single cell containing Python code for object detection. The code includes imports for `object\_detection` and `os`, a command to list pip packages, and a series of commands to download a pretrained model from Google Cloud Storage, extract it, and move files. A progress bar indicates the extraction process is at 100% completion. Below the cell, a terminal window shows the command history and the final message "1 file(s) moved.".

```
In [5]: import object_detection  
In [6]: !pip list  
In [11]: #the code will download the pretrained model, extract it to a local directory  
#and then move it to the appropriate location.  
if os.name == "posix":  
    !gdown --id PRETRAINED_MODEL_URL  
    !tar -xvf PRETRAINED_MODEL_NAME+"tar.gz" [paths["PRETRAINED_MODEL_PATH"]]  
    !cd [paths["PRETRAINED_MODEL_PATH"]] && tar -xvf {PRETRAINED_MODEL_NAME}.tar.gz  
if os.name == "nt":  
    !gdown --id PRETRAINED_MODEL_URL  
    !move PRETRAINED_MODEL_NAME+".tar.gz" [paths["PRETRAINED_MODEL_PATH"]]  
    !cd [paths["PRETRAINED_MODEL_PATH"]] && tar -xvf {PRETRAINED_MODEL_NAME}.tar.gz  
8% [.....] 167118 / 20515344  
19% [.....] 4093656 / 20515344  
31% [.....] 6455296 / 20515344  
42% [.....] 8814592 / 20515344  
55% [.....] 11345920 / 20515344  
66% [.....] 13697014 / 20515344  
79% [.....] 16220100 / 20515344  
90% [.....] 18472968 / 20515344  
100% [.....] 20515344 / 20515344  
1 file(s) moved.  
x ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu_8/checkpoint/  
x ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu_8/checkpoint/  
x ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu_8/checkpoint/ckpt-0,data-00000..00001  
x ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu_8/checkpoint
```

## 4.10 Code 6

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** Training & Placement Cell App → 2. Training and Detection - Jupyter
- URL:** localhost:8888/notebooks/2%20Training%20and%20detection.ipynb
- Header:** jupyter 2. Training and Detection Last Checkpoint 07/27/2022 (autosaved)
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help
- Cell 1 (Code):**

```
In [1]: #The code is used to create a list of labels and their corresponding ids.  
#The code above opens the file with the name LABELMAP,  
#which contains all the label names and their respective ids
```
- Cell 2 (Code):**

```
In [6]: labels = [{"name": "licence", "id": 1}]  
with open("labelmap.txt", "w") as f:  
    for label in labels:  
        f.write(item + "\n")  
        f.write(item + "\n", format(label["name"]))  
        f.write(item + "\n", format(label["id"]))  
        f.write(item + "\n")
```
- Section 3:** 3. Create TF records
- Cell 3 (Code):**

```
In [7]: # OPTIONAL: IF RUNNING ON COLAB  
# os.path.exists(IMAGE_PATH), 'archive_path', 'archive.tar.gz')  
# if os.path.exists(ARCHIVE_FILES):  
#     !tar -xvf ARCHIVE_FILES
```
- Cell 4 (Code):**

```
In [13]: if not os.path.exists(file["TF_RECORD_SCRIPT"]):  
    !git clone https://github.com/nickmochack/GenerateTFRecord {paths["SCRIPTS_PATH"]}
```
- Cell 5 (Code):**

```
In [14]: python [file["TF_RECORD_SCRIPT"]] >x (os.path.join(paths["IMAGE_PATH"], 'train')) -> [files["LABELMAP"]]-> (os.path.join(paths["TEST_PATH"], 'test')) -> [files["LABELMAP"]]-> (os.path.join(paths["TEST_PATH"], 'test'))
```
- Output:**

```
Successfully created the TRecord file: Tensorflow/workspace/annotations/train.record  
Successfully created the TRecord file: Tensorflow/workspace/annotations/test.record
```

## Automatic Number Plate Detection

### 4.11 Code 7

The screenshot shows a Jupyter Notebook interface with two code cells. The first cell contains Python code for copying files from a 'PRETRAINED\_MODEL\_PATH' to a 'CHECKPOINT' folder. The second cell is titled '5. Update Config For Transfer Learning' and contains code to import TensorFlow, config\_util, pipeline\_pb2, and google.protobuf.text\_format, then create a config object from a pipeline file.

```
In [15]: if os.name == 'posix':
    if os.path.exists(paths['PRETRAINED_MODEL_PATH']):
        PRETRAINED_MODEL_NAME, 'pipeline.config')) (os.path.join(paths['CHECKPOINT'],
if os.name == 'nt':
    copy(os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'pipeline.config')) (os.path.join(paths['CHECKPOINT'],
1 file(s) copied.

5. Update Config For Transfer Learning

In [7]: import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format

In [8]: config = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])

In [18]: config
```

### 4.12 Code 8

The screenshot shows a Jupyter Notebook interface with several code cells. Cells 9 through 14 are part of a larger block of code for reading a pipeline configuration file and merging it with a local one. Cells 15 through 19 show the configuration being modified for a specific dataset. Cell 20 is titled '6. Train the model'. Cells 22, 23, and 24 show the command to run the training script with specific arguments.

```
In [9]: #code them reads in the pipeline file free disk using gfile and stores it as a string called proto_str
pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "r") as f:
    proto_str = f.read()
text_format.Merge(proto_str, pipeline_config)

In [10]: pipeline_config.model.ssd.num_classes = len(labels)
pipeline_config.train_config.batch_size = 1
pipeline_config.train_config.fine_tune_checkpoint = os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'checkpoint')
pipeline_config.train_config.fine_tune_checkpoint_type = "detection"
pipeline_config.train_input_reader.label_map_path = files['LABELMAP']
pipeline_config.train_input_reader.tf_record_input_reader.input_path[:] = [os.path.join(paths['ANNOTATION_PATH'], 'train.record')]
pipeline_config.eval_input_reader[0].label_map_path = files['LABELMAP']
pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[:] = [os.path.join(paths['ANNOTATION_PATH'], 'test.record')]

In [11]: config_text = text_format.MessageToString(pipeline_config)
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "wb") as f:
    f.write(config_text)

6. Train the model

In [22]: TRAINING_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection', 'model_main_tf2.py')

In [23]: command = "python {} --model_dir={} --pipeline_config_path={} --num_train_steps=10000".format(TRAINING_SCRIPT, paths['CHECKPOINT'],
...

In [24]: print(command)
```

## Automatic Number Plate Detection

### 4.13 Code 9

```
In [12]: import os
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
from object_detection.utils import config_util

In [13]: #gpu = tf.config.list_physical_devices('GPU')
# if gpu:
#   print('yes')

In [14]: # to prevent gpu complete consumption
gpu = tf.config.list_physical_devices('GPU')
if gpu:
    try:
        tf.config.experimental.set_virtual_device_configuration(
            gpu[0], [tf.config.experimental.VirtualDeviceConfiguration(memory_limit=5120)])
    except RuntimeError as e:
        print(e)

In [15]: # Load pipeline config and build a detection model
config = config_util.get_configs_from_pipeline_file(file['PIPELINE_CONFIG'])
detection_model = model_builder.build(model_config=config['model'], is_training=False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-11')).expect_partial()

@tf.function
def detect_fn(image):
```

### 4.14 Code 10

```
In [14]: # Load pipeline config and build a detection model
config = config_util.get_configs_from_pipeline_file(file['PIPELINE_CONFIG'])
detection_model = model_builder.build(model_config=config['model'], is_training=False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-11')).expect_partial()

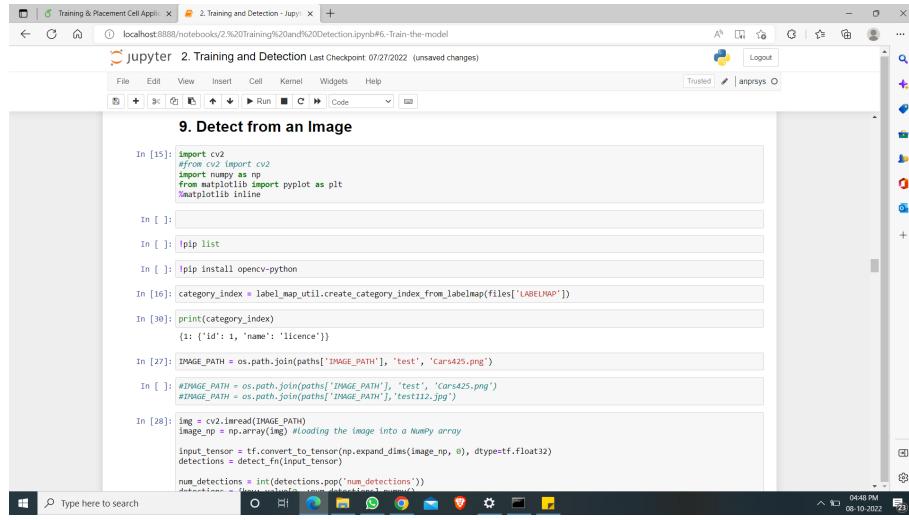
@tf.function
def detect_fn(image):
    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
    prediction_dict.pop('detection_classes')
    detections = detection_model.postprocess(prediction_dict, shapes)
    detections = postprocess_function(detections)
    detections = detections['detection_boxes'].numpy()
    detections = detections.tolist()
    return detections

# the result of this prediction is passed into a post-processing function which will combine all
# of these predictions together and return them as an array of shape ids
```

```
In [15]: import cv2
# from cv2 import cv2
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

## Automatic Number Plate Detection

### 4.15 Code 11



The screenshot shows a Jupyter Notebook interface with the title "jupyter 2. Training and Detection". The notebook has a single cell containing Python code for loading an image, creating a TensorFlow tensor from it, and performing detections. The code is as follows:

```
In [15]: import cv2
from tensorflow import keras
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline

In [16]: category_index = label_map_util.create_category_index_from_labelmap(file["LABELMAP"])

In [17]: print(category_index)
{'1': {'id': 1, 'name': 'licence'}}
```

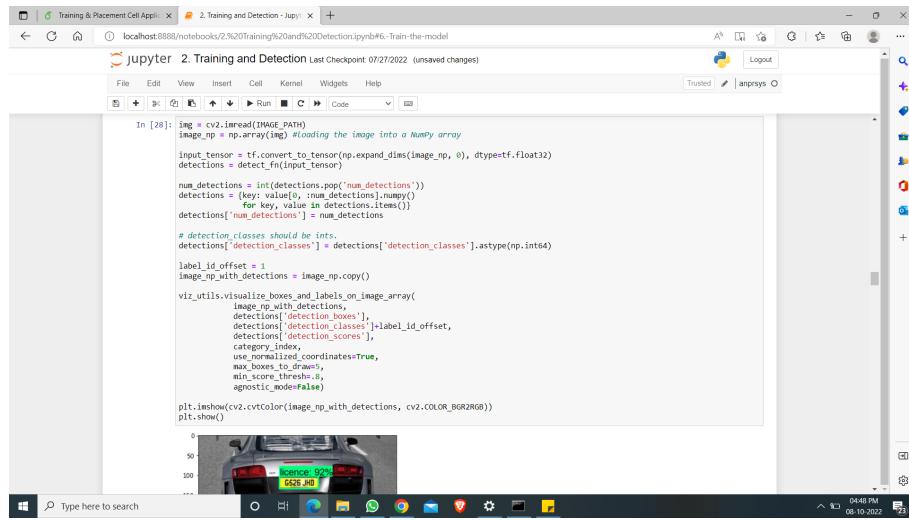
In [27]: IMAGE\_PATH = os.path.join(paths['IMAGE\_PATH'], 'test', 'Cars425.png')

In [28]: #IMAGE\_PATH = os.path.join(paths['IMAGE\_PATH'], 'test', 'Cars425.png')
#IMAGE\_PATH = os.path.join(paths['IMAGE\_PATH'], 'test112.jpg')

In [28]: img = cv2.imread(IMAGE\_PATH)
image\_np = np.array(img) #loading the image into a NumPy array
input\_tensor = tf.convert\_to\_tensor(np.expand\_dims(image\_np, 0), dtype=tf.float32)
detections = detect\_fn(input\_tensor)

num\_detections = int(detections.pop('num\_detections'))
detections['num\_detections'] = num\_detections

### 4.16 Code 12



The screenshot shows a Jupyter Notebook interface with the title "jupyter 2. Training and Detection". The notebook has a single cell containing Python code for loading an image, creating a TensorFlow tensor from it, and performing detections. The code is as follows:

```
In [28]: img = cv2.imread(IMAGE_PATH)
image_np = np.array(img) #loading the image into a NumPy array
input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
detections = detections.pop('num_detections')
detections = {key: value[0].numpy() for key, value in detections.items()}
detections['num_detections'] = num_detections

# detection_classes should be int.
detections['detection_classes'] = detections['detection_classes'].astype(np.int64)
label_id_offset = 1
image_np_with_detections = image_np.copy()
viz_utils.visualize_boxes_and_labels_on_image_array(
    image_np_with_detections,
    detections['detection_boxes'],
    detections['detection_classes'],
    detections['detection_scores'],
    category_index,
    use_normalized_coordinates=True,
    max_boxes_to_draw=1,
    min_score_thresh=.8,
    agnostic_mode=False)
plt.imshow(cv2.cvtColor(image_np_with_detections, cv2.COLOR_BGR2RGB))
plt.show()
```

Below the code cell, there is a small image of a car's rear view with a bounding box around the license plate area, and the text "License 029" and "SG26 JRD" displayed.

## Automatic Number Plate Detection

## 4.17 Code 13

```
In [1]: pip install easyocr

In [2]: pip install torch torchvision torchaudio --extra-index-url https://download.pytorch.org/whl/cu113

In [3]: import easyocr

In [4]: detection_threshold = 0.7

In [5]: image = image_np_with_detections
scores = np.array([lambda x: x * detection_threshold, detections['detection_scores']])
boxes = detections['detection_boxes'][:len(scores)]
classes = detections['detection_classes'][:len(scores)]

In [6]: detections['detection_scores']

Out[6]: array([0.91961714, 0.11380354, 0.0897484, 0.0696592, 0.04677545,
       0.03112544, 0.03162478, 0.02527562, 0.02474693, 0.02471384,
       0.02021886, 0.02092078, 0.01816007, 0.01788282, 0.01638636,
       0.0158258, 0.01552578, 0.01522568, 0.01500097, 0.01477597,
       0.0145188, 0.0142477, 0.0144616, 0.01437876, 0.01435081,
       0.01436818, 0.01422996, 0.01379808, 0.01348619, 0.01355332,
       0.01339983, 0.01324411, 0.01305558, 0.0130077, 0.01281225,
       0.01262574, 0.0125745, 0.0124677, 0.01243546, 0.01237335,
       0.01208571, 0.01193847, 0.01183592, 0.01171125, 0.01166235,
       0.01136716, 0.01154521, 0.0113259, 0.01105115, 0.01100333,
       0.0108854, 0.0107554, 0.0106254, 0.01050049, 0.01038069,
       0.01029277, 0.00979685, 0.00969813, 0.00954543, 0.00951533,
       0.00940966, 0.00948257, 0.00943425, 0.0093328, 0.00925469,
       0.0091808, 0.0091454, 0.0091125, 0.00908979, 0.00905999,
       0.0090951, 0.00903164, 0.0090122, 0.00878334, 0.00873905,
       0.00876074, 0.00862203, 0.00848473, 0.00840884, 0.00828762,
       0.00826831, 0.00815299, 0.00805504, 0.00795398, 0.0078274,
```

## 4.18 Code 14

Training & Placement Cell Application

2. Training and Detection - Jupyter Notebook

localhost:8888/notebooks/2%20Training%20and%20Detection.ipynb#6-Train-the-model

jupyter 2. Training and Detection Last Checkpoint: 07/22/2022 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Notebook saved Trusted anprsys O Logout

```
In [15]: width = image.shape[1]
height = image.shape[0]
```

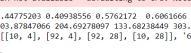
```
In [16]: height
Out[16]: 232
```

```
In [17]: for idx, box in enumerate(boxes):
    print(box)
    print(f'{idx}: {width}, {height}, {height}, {width}')
    print(rois)
    region = Image[int(rois[0]):int(rois[1]), int(rois[2]):int(rois[3])]
```

```
reader = easyocr.Reader(['en'])
ocr_result = reader.readtext(region)
print(ocr_result)
plt.imshow(cv2.cvtColor(region, cv2.COLOR_BGR2RGB))
```

CUDA not available - defaulting to CPU. Note: This module is much faster with a GPU.

```
[0,44775203 0.4993856 0.5762172 0.6081666]
[103,87047866 204,69778997 133,68238449 303,68330059]
[[[10, 4], [92, 4], [92, 28], [10, 28]], [6526 3HD, 0, 6684800252219833]]
```

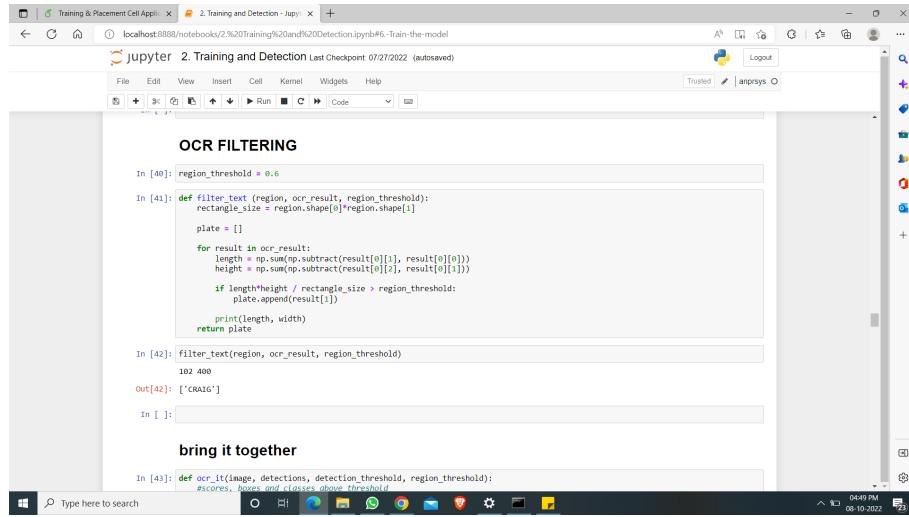


```
In [36]: for result in ocr_result:
    print(np.sum(np.subtract(result[0][2], result[0][1])))
    print(result[1])
```

```
24
6526 JHD
```

## Automatic Number Plate Detection

### 4.19 Code 15



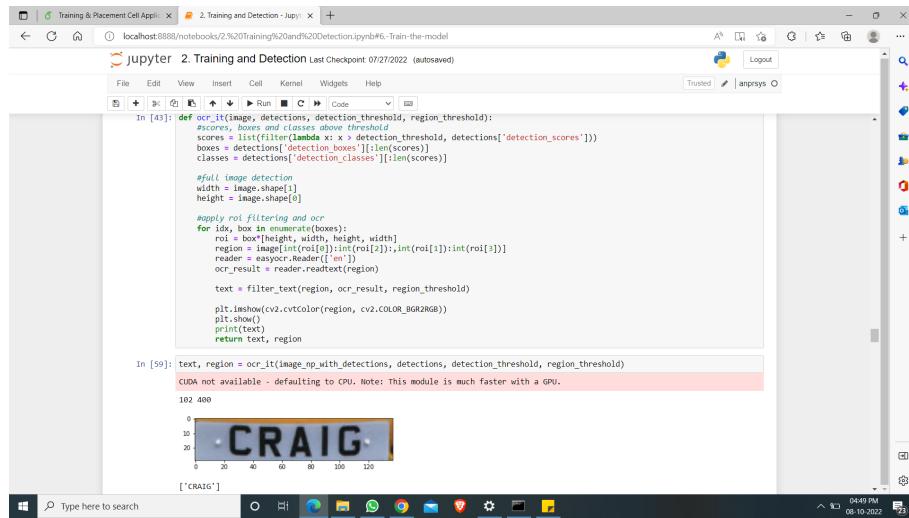
The screenshot shows a Jupyter Notebook interface with a single cell containing Python code. The code defines a function `filter_text` that takes a region, an OCR result, and a region threshold. It calculates the length and width of each result and appends it to a list if the aspect ratio is greater than the threshold. The code then prints the length and width of the first result and returns the list. A second cell shows the output: a list containing the string 'CRAIG'.

```
In [40]: region_threshold = 0.6
In [41]: def filter_text(region, ocr_result, region_threshold):
    rectangle_size = region.shape[0]*region.shape[1]
    plate = []
    for result in ocr_result:
        length = np.sum(np.subtract(result[0][1], result[0][0]))
        height = np.sum(np.subtract(result[0][2], result[0][1]))
        if length/height / rectangle_size > region_threshold:
            plate.append(result[1])
    print(length, width)
    return plate
In [42]: filter_text(region, ocr_result, region_threshold)
102 400
Out[42]: ['CRAIG']
In [ ]:
```

bring it together

```
In [43]: def ocr_it(image, detections, detection_threshold, region_threshold):
    scores = detections['detection_scores'][boxes_and_classes_above_threshold]
    boxes = detections['detection_boxes'][len(scores)]
    classes = detections['detection_classes'][len(scores)]
    width = image.shape[1]
    height = image.shape[0]
    for id, box in enumerate(boxes):
        roi = box[height, width, height, width]
        region = image[int(roi[0]):int(roi[2]):int(roi[1]):int(roi[3])]
        reader = easyocr.Reader(['en'])
        ocr_result = reader.readText(region)
        text = filter_text(region, ocr_result, region_threshold)
        plt.imshow(cv2.cvtColor(region, cv2.COLOR_BGR2RGB))
        plt.show()
        print(text)
    return text, region
In [ ]:
```

### 4.20 Code 16



The screenshot shows a Jupyter Notebook interface with a single cell containing Python code. The code defines a function `ocr_it` that takes an image, detections, a detection threshold, and a region threshold. It filters the detections based on the detection threshold, extracts the boxes and classes above the threshold, and then processes each box to extract text. For each box, it creates a region of interest (ROI) from the image, reads the text using an EasyOCR reader, and then applies the `filter_text` function to the ROI. The extracted text and the ROI are returned. A note at the bottom indicates that CUDA is not available and defaulting to CPU. The output shows the text 'CRAIG' and the corresponding ROI image.

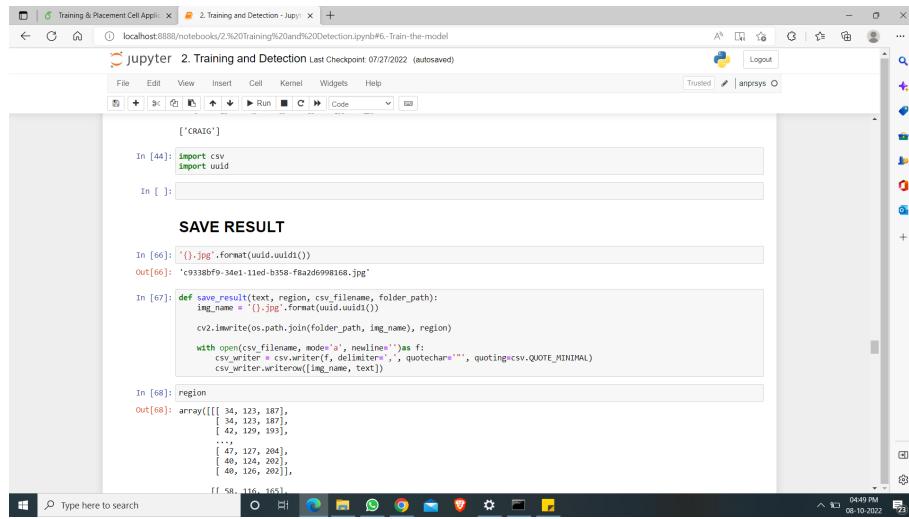
```
In [43]: def ocr_it(image, detections, detection_threshold, region_threshold):
    scores = detections['detection_scores'][boxes_and_classes_above_threshold]
    boxes = detections['detection_boxes'][len(scores)]
    classes = detections['detection_classes'][len(scores)]
    width = image.shape[1]
    height = image.shape[0]
    for id, box in enumerate(boxes):
        roi = box[height, width, height, width]
        region = image[int(roi[0]):int(roi[2]):int(roi[1]):int(roi[3])]
        reader = easyocr.Reader(['en'])
        ocr_result = reader.readText(region)
        text = filter_text(region, ocr_result, region_threshold)
        plt.imshow(cv2.cvtColor(region, cv2.COLOR_BGR2RGB))
        plt.show()
        print(text)
    return text, region
In [ ]:
```

CRAIG

```
[CRAIG]
```

## Automatic Number Plate Detection

### 4.21 Code 17



The screenshot shows a Jupyter Notebook interface with two code cells. The first cell contains:

```
In [44]: import csv  
import uuid
```

The second cell contains:

```
In [45]: SAVE RESULT
```

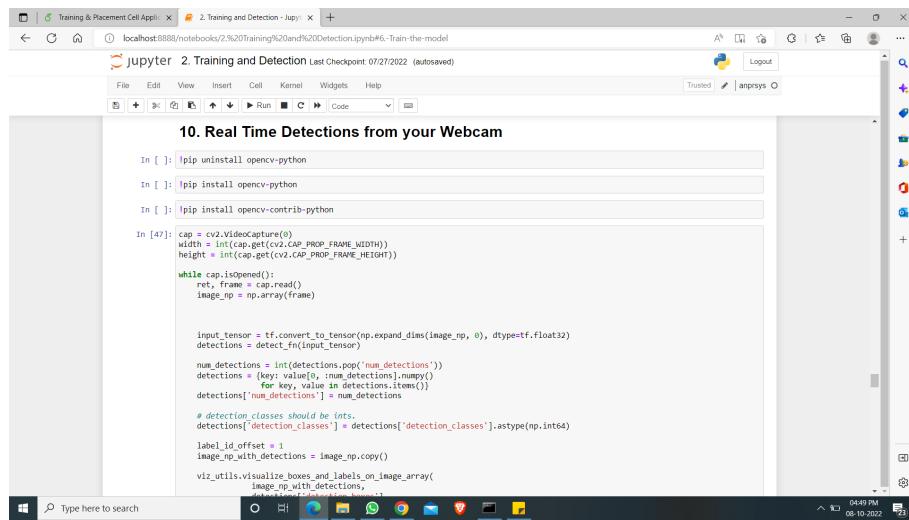
```
In [46]: ('{}-{}').format(uuid.uuid1())  
Out[46]: 'c9338bf9-34e1-11ed-b358-f8a2d6998168.jpg'
```

```
In [47]: def save_result(text, region, csv_filename, folder_path):  
    img_name = '{}.jpg'.format(uuid.uuid1())  
    cv2.imwrite(os.path.join(folder_path, img_name), region)  
    with open(csv_filename, mode='a', newline='') as f:  
        csv_writer = csv.writer(f, delimiter=',', quotechar='"', quoting=csv.QUOTE_MINIMAL)  
        csv_writer.writerow([img_name, text])
```

```
In [48]: region
```

```
Out[48]: array([[ 34, 123, 187],  
                 [ 34, 123, 187],  
                 [ 42, 129, 193],  
                 ...,  
                 [ 47, 127, 204],  
                 [ 47, 127, 204],  
                 [ 48, 126, 202],  
                 [ 58, 116, 165],
```

### 4.22 Code 18



The screenshot shows a Jupyter Notebook interface with three code cells. The first cell contains:

```
In [1]: pip uninstall opencv-python
```

```
In [2]: pip install opencv-python
```

```
In [3]: pip install opencv-contrib-python
```

The second cell contains:

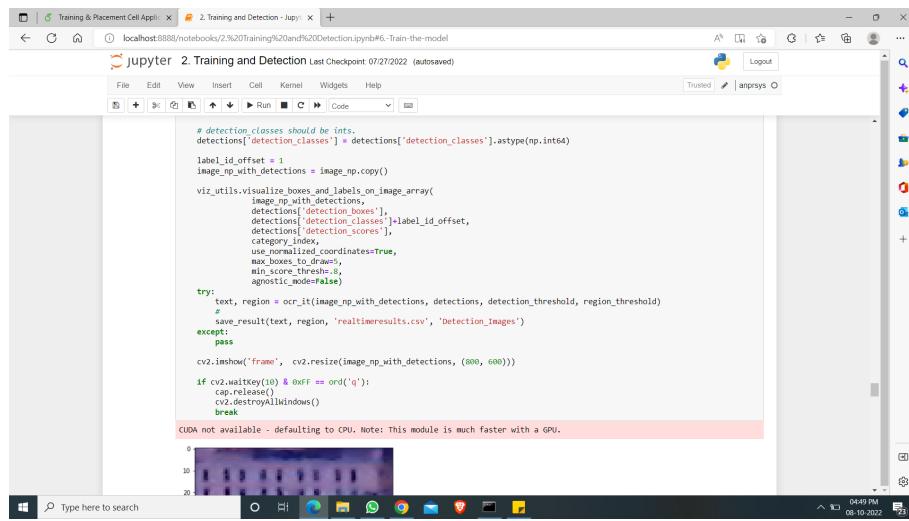
```
In [47]: cap = cv2.VideoCapture(0)  
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))  
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))  
  
while cap.isOpened():  
    ret, frame = cap.read()  
    image_np = np.array(frame)  
  
    input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)  
    detections = detect_fn(input_tensor)  
  
    num_detections = int(detections.pop('num_detections'))  
    detections = {key: value[0, :num_detections].numpy()  
                 for key, value in detections.items()}  
    detections['num_detections'] = num_detections  
  
    # detection_classes should be ints.  
    detections['detection_classes'] = detections['detection_classes'].astype(np.int64)
```

The third cell contains:

```
label_id_offset = 1  
image_np_with_detections = image_np.copy()  
viz_utils.visualize_boxes_and_labels_on_image_array(  
    image_np_with_detections,  
    detections['detection_boxes'],  
    detections['detection_classes'],  
    detections['detection_scores'],  
    category_index,  
    use_normalized_coordinates=True,  
    max_boxes_to_draw=200,  
    min_score_thresh=.3, # change to 0.4  
    agnostic_mode=False)
```

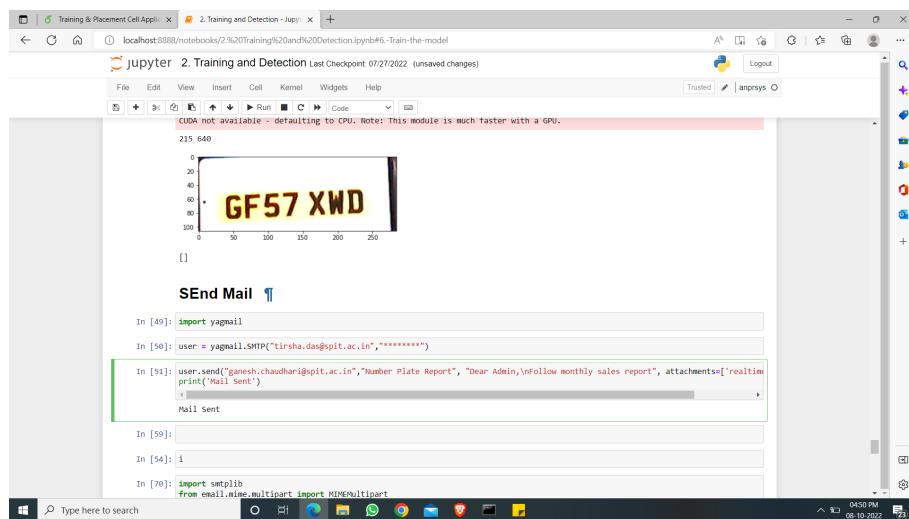
## Automatic Number Plate Detection

### 4.23 Code 19



```
# detection_classes should be ints.
detections["detection_classes"] = detections["detection_classes"].astype(np.int64)
label_id_offset = 1
image_np_with_detections = image_np.copy()
viz_utils.visualize_boxes_and_labels_on_image_array(
    image_np_with_detections,
    detections["detection_boxes"],
    detections["detection_classes"]+label_id_offset,
    detections["detection_scores"],
    category_index,
    use_normalized_coordinates=True,
    max_boxes_to_draw=1000,
    min_score_thresh=.8,
    agnostic_mode=False)
try:
    text, region = ocr_it(image_np_with_detections, detections, detection_threshold, region_threshold)
    save_result(text, region, 'realtimeresults.csv', 'Detection_Images')
except:
    pass
cv2.imshow('frame', cv2.resize(image_np_with_detections, (800, 600)))
if cv2.waitKey(1) & 0xFF == ord('q'):
    cap.release()
    cv2.destroyAllWindows()
    break
CUDA not available - defaulting to CPU. Note: This module is much faster with a GPU.
```

### 4.24 Code 20



```
In [49]: import yagmail
In [50]: user = yagmail.SMTP("tirsha.das@spit.ac.in","*****")
In [51]: user.send("ganesh.chaudhari@spit.ac.in","Number Plate Report", "Dear Admin,\nFollow monthly sales report", attachments=[realtime])
print("Mail Sent")
Mail Sent
In [59]:
In [54]: i
In [70]: import smtplib
from email.mime.multipart import MIMEMultipart
```

## 5 Test Cases

Table 6.1: Test Case

Test Case ID	Test Case Name	Test Data	Expected Output	Actual Output	Result
1	Number Plate detection From Image	Give Proper Image	Detect Number from license plate	Detect Number from license plate	Pass
2	Number Plate detection From Video	Proper Video Footage	Detect Number from license plate	Detect Number from license plate	Pass
3	Number Plate detection real time	Proper Video Footage	Detect Number from license plate	Detect Number from license plate	Pass
4	Rotated Number Plate	License plate is rotated	Detect Proper Number from license plate	Detects Incorrect	Fail

## 6 Limitations

- Privacy -

It is true that records and images are stored and kept but it leads to some issues related to privacy. Usually, people are worried that the information of their whereabouts might be misused which are recorded in those footages. It can get into wrong hands or be subject to data thefts.

But experts claim that automatic number plate recognition doesn't infringe the privacy of anyone. License plate numbers are all police needs as they are issued for public safety. In addition, this system checks every plate automatically and it doesn't involve any discrimination. The agencies which are responsible are considering these privacy concerns and ensure the safety of people on the roads.

- Extreme weather can affect the accuracy -

Hindrances and extreme weather conditions can affect the accuracy of automatic license plate recognition software. Manned surveillance would be required because automatic security systems might not work.

## 7 Future Enhancements

- The number plate recognition system can be enhanced to record all types of number plates whatever their size and font type may be
- Could Detect the fake Number plate and take the image of the driver and store it in the database.
- The number plate recognition system can be enhanced to detect rotated number plate
- Could detect the vehicle which run fast if they are breaking the traffic rules than we could put fine on the vehicle driver

## 8 User Manual

### Run the Code

Go to folder path and then type the commands

If you want to detect the number from an image then all you have to do is add the image path and run the code to detect numbers from license plate

If you want to detect the number from an video then all you have to do is add the video path and run the code to detect numbers from license plate.

If you want to detect the number of license in real time environment the system should have connection to camera and run the code, the camera will start to detect

## 9 Bibliography

### 9.1 Web References

- [1.] <http://developer.android.com/docs/>
- [2.] <https://firebase.google.com/docs>
- [3.] <https://www.youtube.com/user/Firebase>
- [4.] <https://stackoverflow.com/>
- [5.] <https://www.draw.io/>
- [6.] <https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/>
- [7.] <https://www.geeksforgeeks.org/>