Installation Guide

# Requirements

## Hardware requirements:

* Laptop/Computing device
* GPU (Graphic Processor Unit)
* RAM (above 8GB)
* Camera setup / Webcam

## Software requirements:

* Python
* Jupyter notebook
* VISUAL STUDIO CODE
* OpenCV
* Tensorflow
* NVIDIA CUDA
* NVIDIA CUDNN
* EASYOCR

## Elaboration on the tools used:

### Hardware:

1. Laptop/Computing device

A device to research and perform our coding and implementation. The main codes and software will be run through this device. Also, all the outputs that should come out on the database will be stored in its drive.

1. GPU (Graphic Processor Unit)

GPU is required for Machine learning that will be done in this project. CPU can also be utilized if GPU is not available. But definitely will recommend having a GPU as it will save a lot of time on machine learning.

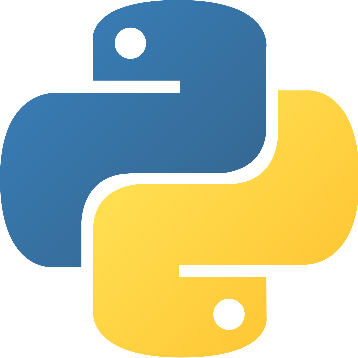
1. RAM (above 8GB)

RAM will determine the process faster on certain types of operations. Having more RAM will increase the performance of the project.

1. Camera setup / Webcam

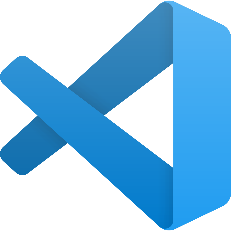
This setup is required for the input of data. The live video feed will be using the camera to run and process the displayed characters, which in our case is the license plate numbers.

### Software:

1. PYTHON

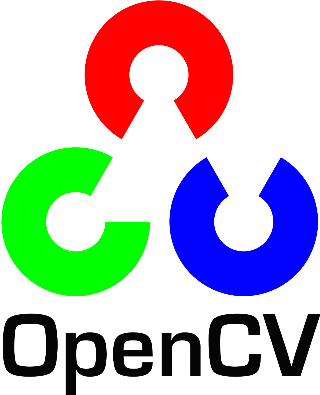
Python is an object-oriented, high-level programming language with dynamic semantics that is construed. Its strong built-in data types, mixed with flexible editing as well as flexible binding consider it very appealing for Faster system Development while also using as a typescript or glue syntax to attach existing modules. Python's convenient, user - friendly language prioritizes readability, lowering the expense of program construction Python includes assistance for components and packages, that promotes methods and practices as well as refactoring.

1. JUPYTER NOTEBOOK

Jupyter Notebook is a free and open-source web application that lets you build and manage data with live script, symmetries, templates, and statistical analysis. Some used are: Data screening and modification, simulation analysis, data analysis, application development, and artificial intelligence.

1. VISUAL STUDIO CODE

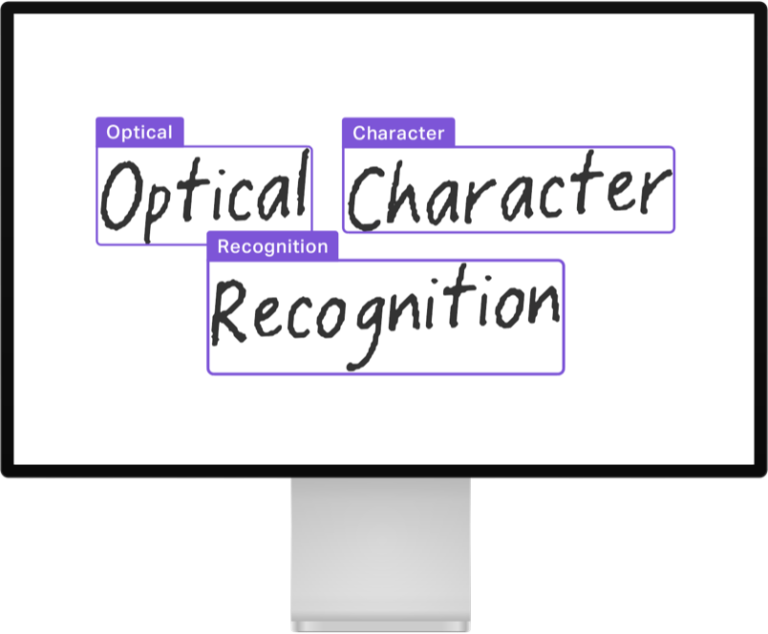
Microsoft's Visual Studio Code is a source-code editor. Syntax editing, debugging, intelligent code matching, fragments, bug fixes, and integrated Git are among the few of many features.

1. OPENCV

OpenCV is a massive open-source framework for machine learning, algorithms, and image recognition, and it currently performs a significant part in real time functions which is critical in modern technologies.

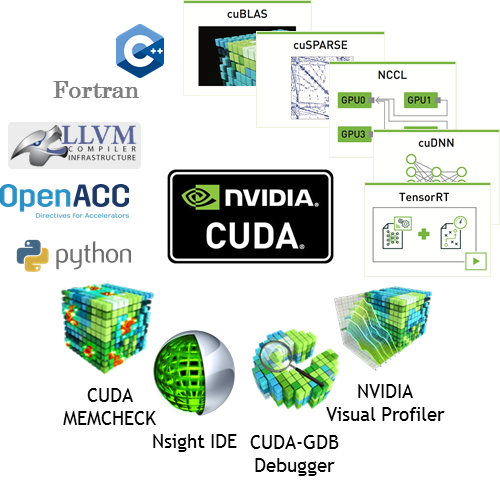
1. TENSORFLOW

TensorFlow is a machine learning development kit which is an open-source software. It could be used for a variety of applications, though it mainly focuses on deep learning models training and testing. Tensorflow is an expressive mathematical library built on datatypes and distinct algorithms.

1. EASYOCR

EasyOCR is a Python tool which enables users to translate images into textual format. This is perhaps the best method of implementing OCR as it supports more than 70 languages, such as English, Hindi, Devanagari, Chinese and several more. EasyOCR is a developed by the Jaided AI business.

1. NVIDIA GPU

Nvidia GPUs widely popular for machine learning and rapid analysis because of Nvidia's CUDA API, that allow developers to make use of the larger multiple cores available in GPUs to use BLAS operations, that are highly needed in machine learning models.

1. CUDA

Nvidia's CUDA is a serial processing platform and application programming interface framework. It helps programmers and developers to use a CUDA-enabled graphics card for specific processing – a process known as GPGPU.

1. CUDNN

CUDA Deep Neural Network (cuDNN) by NVIDIA is a primate package using graphic processor for deep learning techniques. This provides amazing optimized variants of procedures that are often encountered in DNN algorithms.

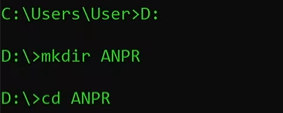
# Installing ANPR System

First, we create our virtual environment where we will be working our project.

On our D drive we open a new folder and name it ANPR through ‘mkdir’.

Then we create a new virtual environment there and name it ‘anprs’.

To activate our virtual environment we type “.\anprs\scripts\activate “

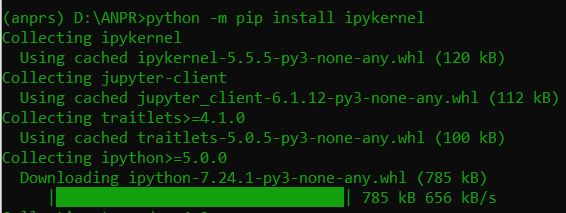


We will also be needing ipykernel.

To install that we write the code “python -m pip install ipykernel”

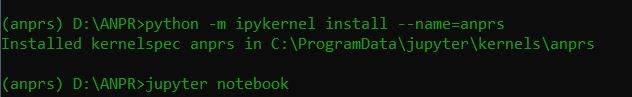
Now to associate our virtual environment (anprs) with our kernel we write the code

“python -m ipykernel install –name=anprs”.

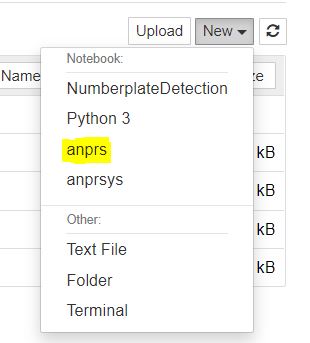


This will allow us to use our virtual environment inside Jupyter Notebook.

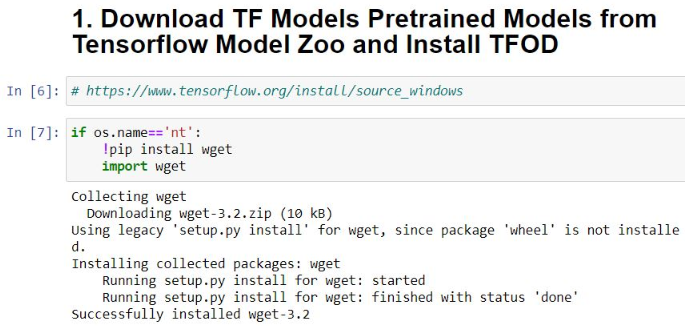
Consequently, we will be using Jupyter Notebook as our main IDE to perform tasks. To open Jupyter notebook we just type “Jupyter notebook” on cmd (Figure). Accordingly, the main required dependencies are up and running.



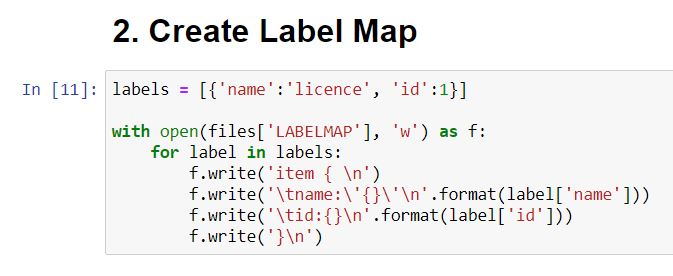
New kernel will be displayed as:



Now from the tensorflow model zoo we install and import dependencies into our TFOD (Tensor Flow Object Detection) which we call as transfer learning. We fine tune the pre trained model to our preference. The “wget” will grub our trained models. The second code will clone the tensorflow zoo official models into our tensorflow model repository folder.



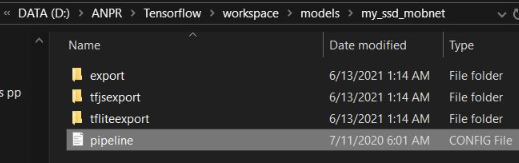
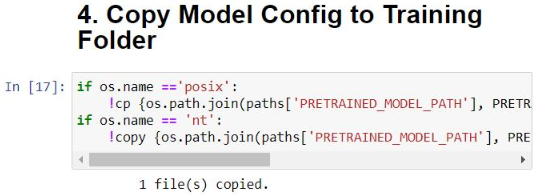
In a label map we add the name of the label from the annotation of the dataset we placed in our ‘train’ folder. After we successfully run the code, it creates a file named ‘label\_map’ inside the workspace folder.



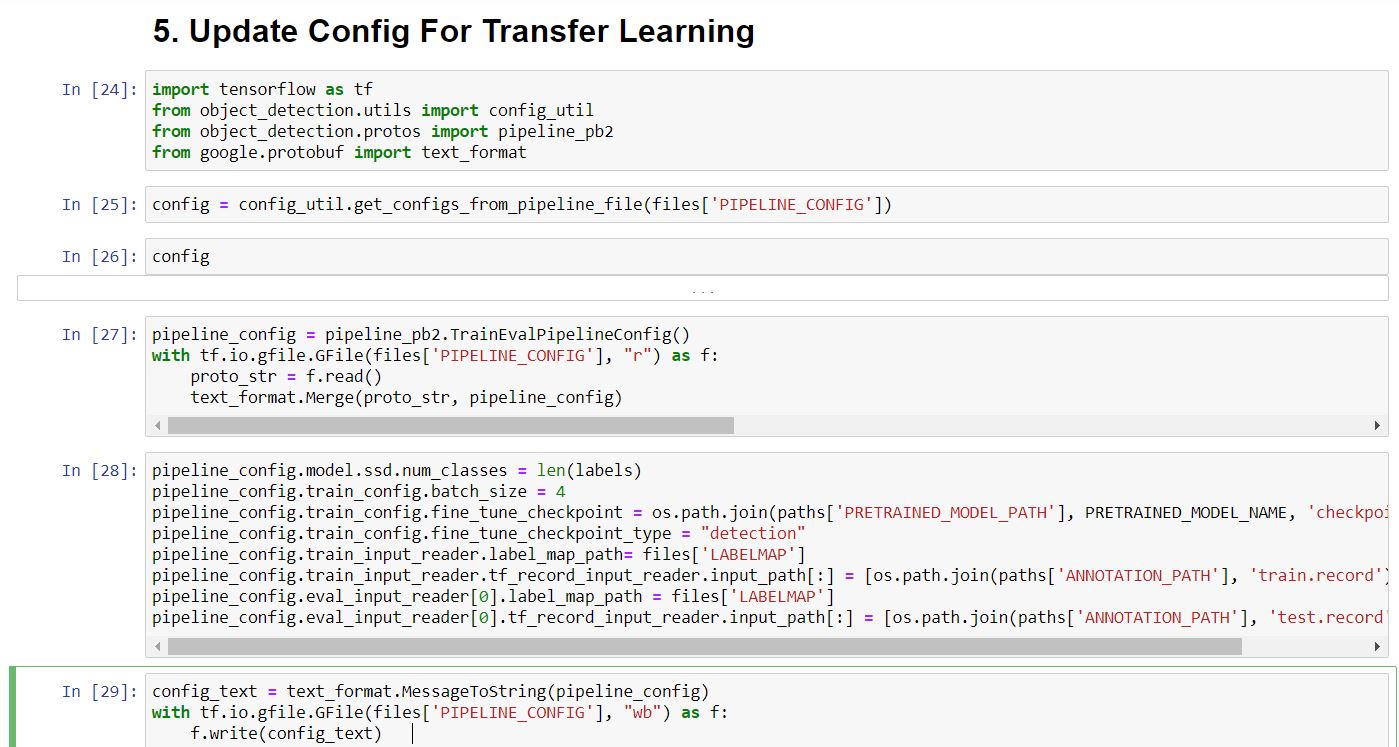
Finally, if the code runs successfully (Figure ), tensorflow records ‘train.record’ and ‘test.record’ gets created in the same folder as ‘label\_map’ was created.



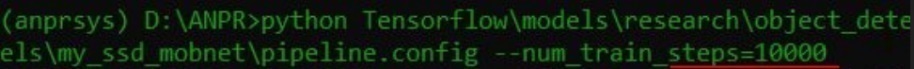
The ‘pipeline.config’ file is copied into our path to train out model.

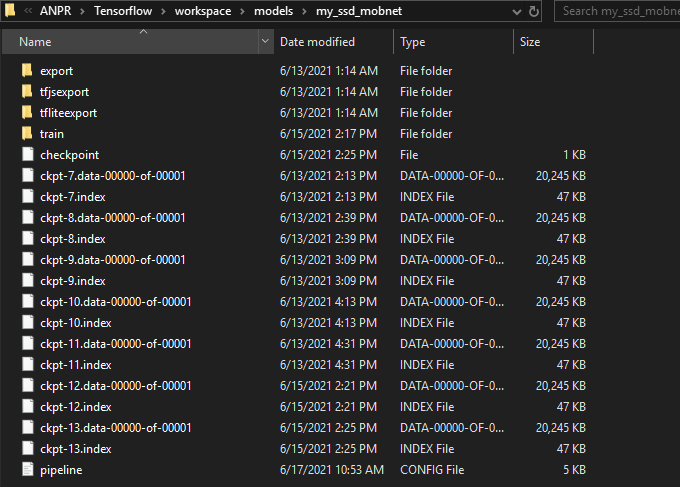


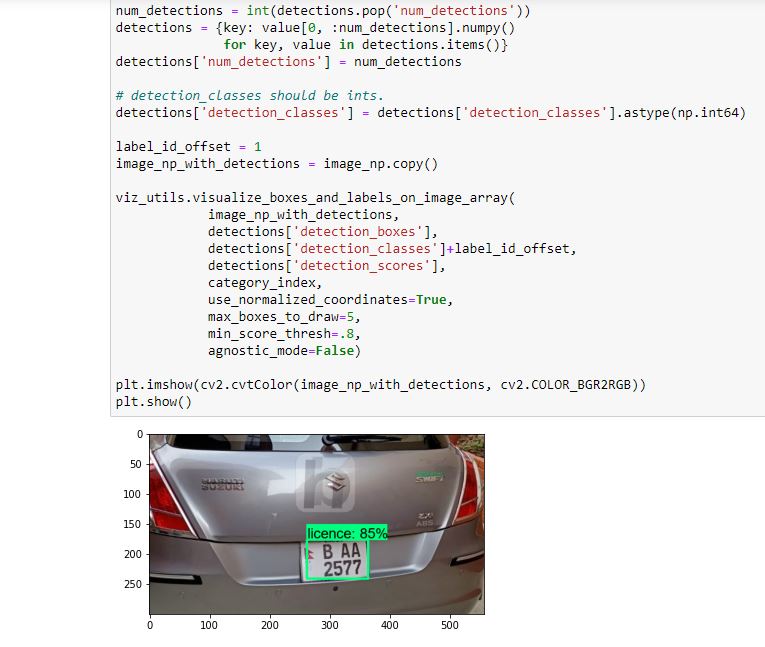
In this we import tensorflow from object detection and various other utilities that we require to successfully run our model.



After the cells have been run, the print command generates a code that start training our model. We will be training our model for 10,000 steps.



To check if the model trained successfully, we go into the folder my\_ssd\_mobnet located in ANPR/Tensorflow/workspace/ models/my\_ssd\_mobnet

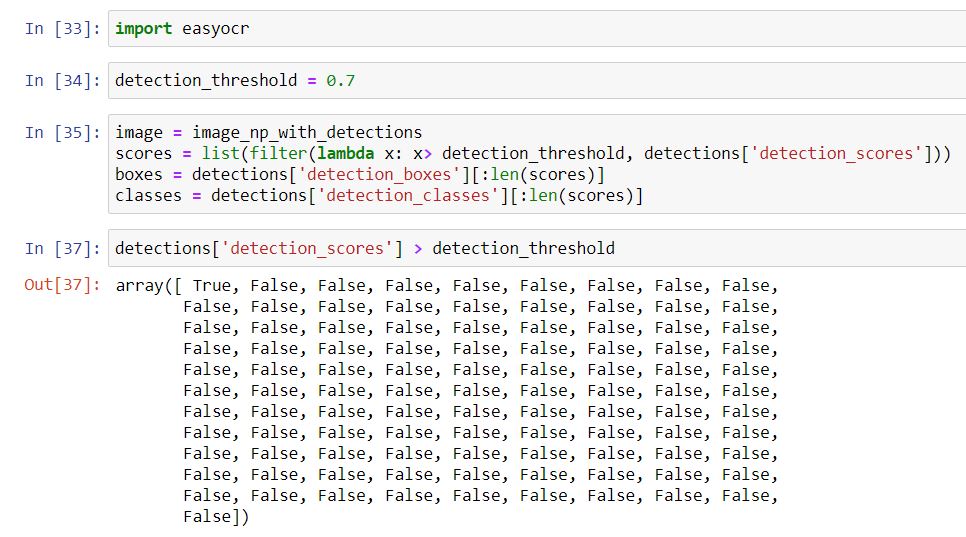
Import all the dependencies first, then the next cell code will import the image through file name that you want to run in our system

The code will open up a connected video camera, in our case connected webcam and from there it will accurately detect the license plate in real time

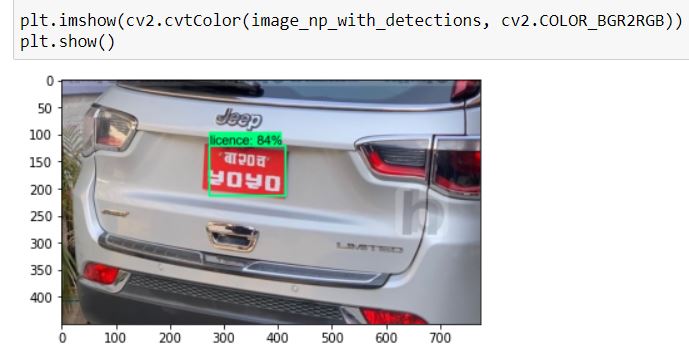
After successfully installing EasyOCR, we now download PyTorch. By going into their website [https://pytorch.org](https://pytorch.org/)

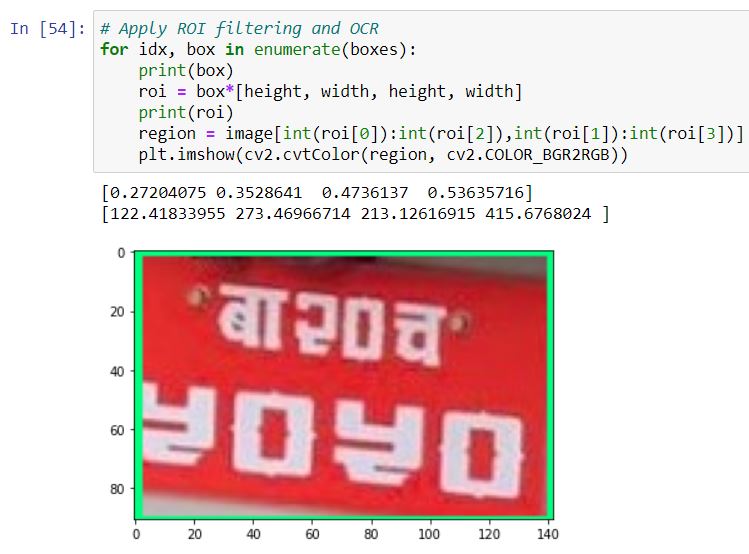
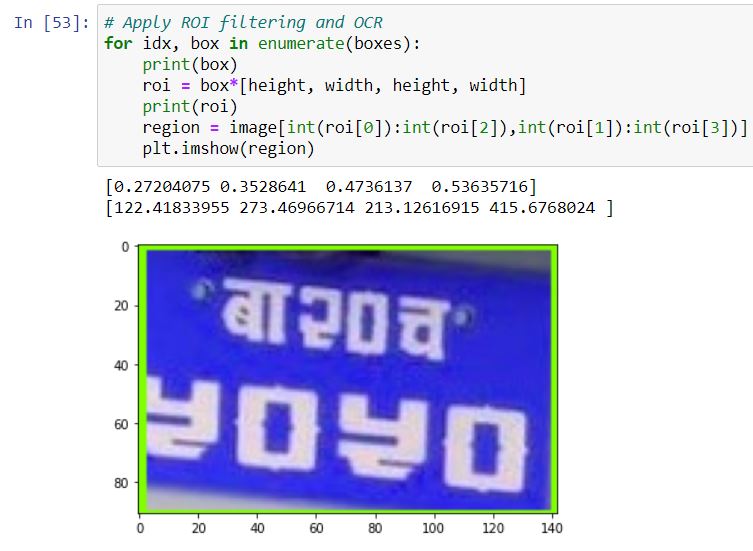


After importing OCR, we set up a variable called ‘detection\_threshold’ where we will render our result. Then we grub our images with the following lines of code. We also extract components like scores, boxes and classes.



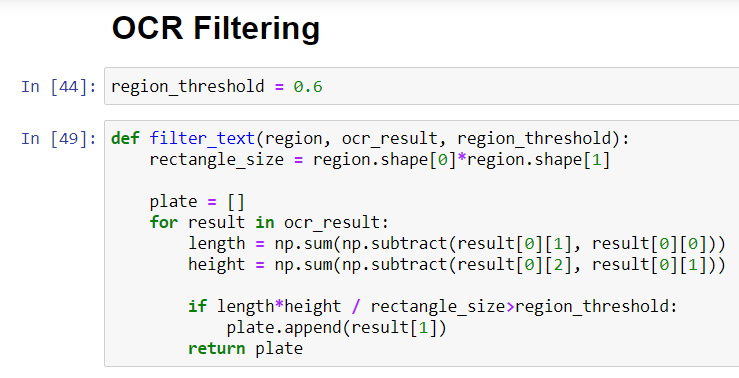
After extracting the height and width of our required ROI (region of interest), we filter and apply OCR

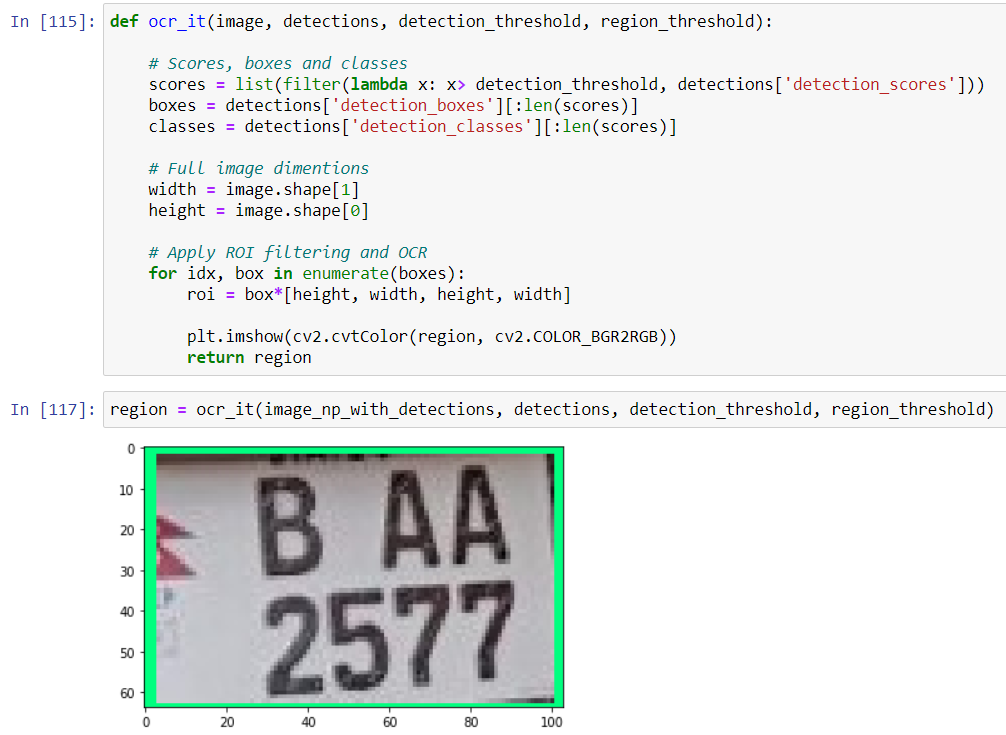


  
After they been multiplied by the co-ordinates of our image

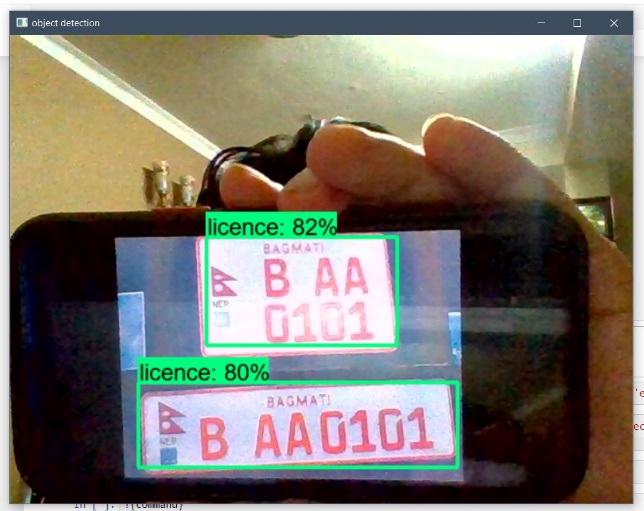
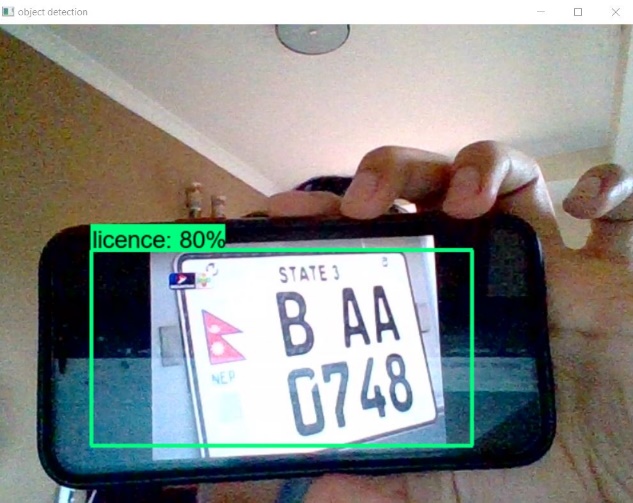
Then we change the colour from BGR to RGB as we can only render the image from matplotlib if it is on RGB format.

If our ROI passes the threshold that we set we will get our text.



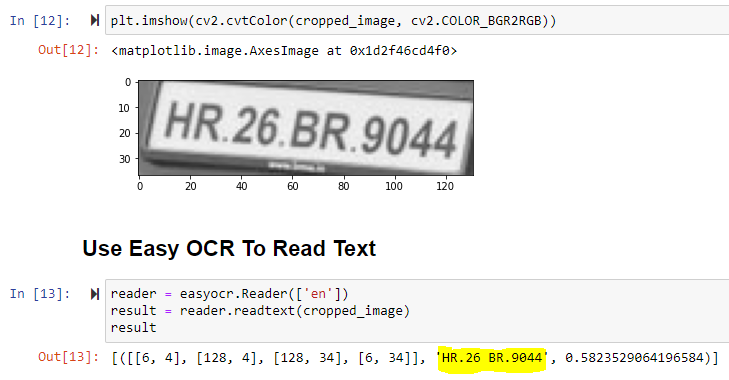
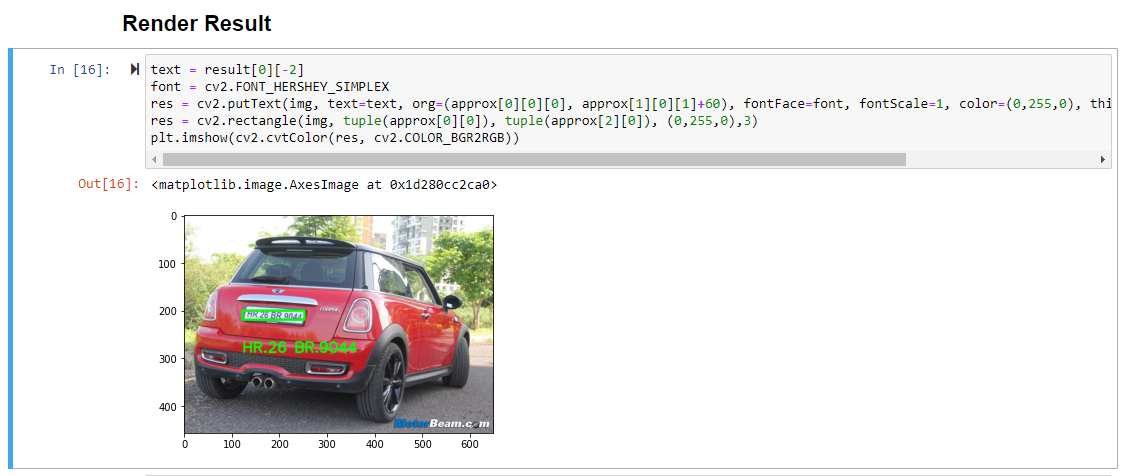
Finally, the real time ANPR detects from image

Also, the code will open up a connected video camera, in our case connected webcam and from there it will accurately detect the license plate in real time. It will extract



To use the OCR function, open the ANPR-OCR.ipynb file on Jupyter notebook

Then run the commands by pressing shift + enter.

OCR will display the numbers from the license plate from an image successfully