AUTOMATIC NUMBER PLATE DETECTION AND RECOGNITION

A Project Report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Sai Tarrun Pitta, 121910302010

Utkarsh MVSS, 121910302051

Mullapudi Keerthi Sri, 121910302064

Under the esteemed guidance of

Dr. Praveen Gupta

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING GITAM

(Deemed to be University)

VISAKHAPATNAM

APRIL 2023

2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM SCHOOL OF TECHNOLOGY

GITAM

(Deemed to be University)

DECLARATION

We, hereby declare that the project report entitled "**AUTOMATIC NUMBER PLATE DETECTION AND RECOGNITION**" is an original work done in the Department of Computer Science and Engineering, GITAM School of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date: 04-04-2023

Registration No(s).	Name(s)	Signature(s)
1. 121910302010	Sai Tarrun Pitta	P. Sai Tarrun
2. 121910302051	Utkarsh MVVS	Utkarsh MVVS
3. 121910302064	Mullapudi Keerthi Sri	Keerthi Sri. M

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM SCHOOL OF TECHNOLOGY

GITAM

(Deemed to be University)

CERTIFICATE

This is to certify that the project report entitled "**Automatic Number Plate Detection And Recognition**" is a bonafide record of work carried out by **Sai Tarrun Pitta (121910302010), Utkarsh MVSS (121910302051) and Mullapudi Keerthi Sri (121910302064)** students submitted in partial fulfillment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering.

Project Guide	**Head of the Department**
07/01/2023 **Dr. Praveen Gupta**	**Dr. R.Sireesha**
Assistant Professor	**Professor**

TABLE OF CONTENTS

1.	Abstract	
2	Introduction	1
3	Literature Review	2
4	Problem Identification & Objectives	3
5	System Methodology	5
6	Overview of Technologies	7
7	Implementation (Coding & Testing)	11
8	Results & Discussions	31
9	Conclusion & Future Scope	32
10	References	33

1. ABSTRACT

The task of real-time number plate detection using a webcam can be approached by combining the strengths of several powerful technologies: TensorFlow, Keras, OpenCV, and EasyOCR.

This project's primary objective is that to build a system that is capable of and recognize real-time number plate scanning, using a webcam as the input source. The number plate area in the image should be detectable by the system and then use OCR to read the characters on the plate.

The procedure's initial stage is to use OpenCV to capture frames from the webcam in real-time. These frames are then processed to detect the number plate region. This can be done using various computer vision techniques, such as edge detection, contour detection, and morphological operations. Once the number plate region has been detected, it can be cropped out and passed on to the next stage.

The next step is to use a deep learning model, implemented using TensorFlow and Keras, to identify the numbers and letters on the licence plate. In order to do this, the model must first be trained on a sizable dataset of annotated number plate images before being applied to the clipped number plate region to carry out character recognition.

Finally, EasyOCR can be used to perform additional character recognition and post-processing on the recognized characters. This can help to improve the accuracy of the system and reduce errors.

Overall, this project involves the integration of several different technologies to build a robust real-time number plate detection system. The outcome is a system that can reliably identify number plates in real-time, making it useful for a range of applications like traffic monitoring, toll collecting, and parking management.

2. INTRODUCTION

Real time pattern recognition is a method used in number plate detection. License plates can be read. Simply put, an ANPR(Automatic number plate detection) camera "pictures" a vehicle's license plate vehicles passing through them. This "picture" is sent to a computer system for detailed examination. ANPR consists of a camera connected to a computer. ANPR when a car passes by digitally "read" a vehicle's license plate (commonly known as a license plate). Cameras located on mobile units, cameras installed or closed-circuit television in automobiles . Data is transformed from digital photos before being processed by the ANPR system. The proposed techniques primarily use edge detection and OCR processes.

Today, owning a car has become a necessity, not just a symbol of luxury. But, when it comes to vehicles, catastrophic situations can occur. Therefore, there are always emergencies. Appropriate measures should be initiated to enhance vehicle safety, security and surveillance to avoid accidents. Useful in the following situations: Instant delivery vehicle, Details with image processing, allowing government agencies to track the location of vehicles, automatically notify the user when the vehicle registers a traffic violation. Gauge is the usage of a GPS-based vehicle tracking device-Global-position-System such as Tracking systems include devices attached to vehicles, Use existing software.

At the base of operations, it aids in tracking the whereabouts of cars. Uses for this base station include example surveillance purposes. It includes maps from Bing Maps, Here Maps, and Google Maps, and more to represent a place. ANPR can be used to store camera-captured images and license texts. A disk that can be configured to store driver pictures.

Infrared lighting is frequently used in systems to allow the camera to take photographs at any time of day. Includes powerful flash. At least one version of his intersection surveillance camera helps illuminate the image and draw the criminal's attention to his mistake. ANPR techniques tend to be region-specific as the plates vary from site to site.

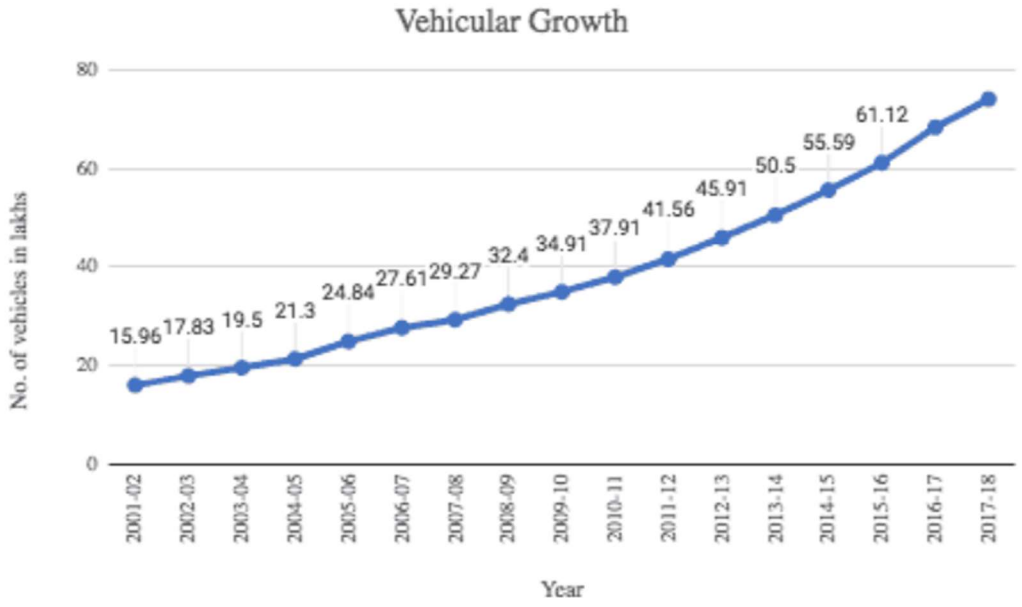


Figure 2.1: Vehicle Growth in India Considering upto year 2018

The proposed system is implemented in Python and tested on a dataset of real-world license plates. The experimental findings demonstrate that the suggested system achieves high accuracy in detection and identification of licence plates. The system is adaptable to a range of applications, including traffic management, parking management, and law enforcement.

This project demonstrates the potential of deep learning-based approaches in solving real-world computer vision tasks and highlights the importance of combining multiple frameworks and libraries to build a robust system.

4. LITERATURE REVIEW

To complete the project, we looked at various research papers on different aspects, such as license plate recognition and recognition, optical character recognition, and YOLO.

J Mukherjee, A Choudhury, and S Roy. [1] They suggested segmenting the numbers to identify each number separately and developing a number plate positioning system primarily for automobiles in West Bengal (India). In this article, we outline a strategy based on the Sobel edge detection method and straightforward, effective morphological operations. Also, he offers a straightforward method for grouping all the letters and numbers found on licence plates. Using histogram equalisation, he attempts to increase the contrast of the binarized image after lowering the noise in the input image. We mostly concentrate on these two actions. The first step is to find the licence plate, and the second is to separate all the letters and digits into individual numbers.

According to the features used in each phase, Du, M. Shehata, and W. Badawy [2] present an overview of the current (Automatic License Plate Recognition) ALPR approaches. We examined the benefits and drawbacks, as well as the recognition outcomes and processing speed. Finally, the ALPR outlook was also demonstrated. Future ALPR research should concentrate on fuzzy character recognition, multi-plate processing, high-resolution plate image processing, video-based ALPR employing temporal information, and multi-style plate identification.

In Tamil Nadu, Professor Anisya S. Mary Joans [3] concentrated on methods for locating and recognising licence plates for vehicles (India). The system, which was created on the basis of digital photographs, is simple to implement in commercial parking systems and can be used to record access to parking services, guarantee the safe use of parking lots, and prevent auto theft. The suggested technique for locating licence plates combines morphological adjustments and area fiducial testing.

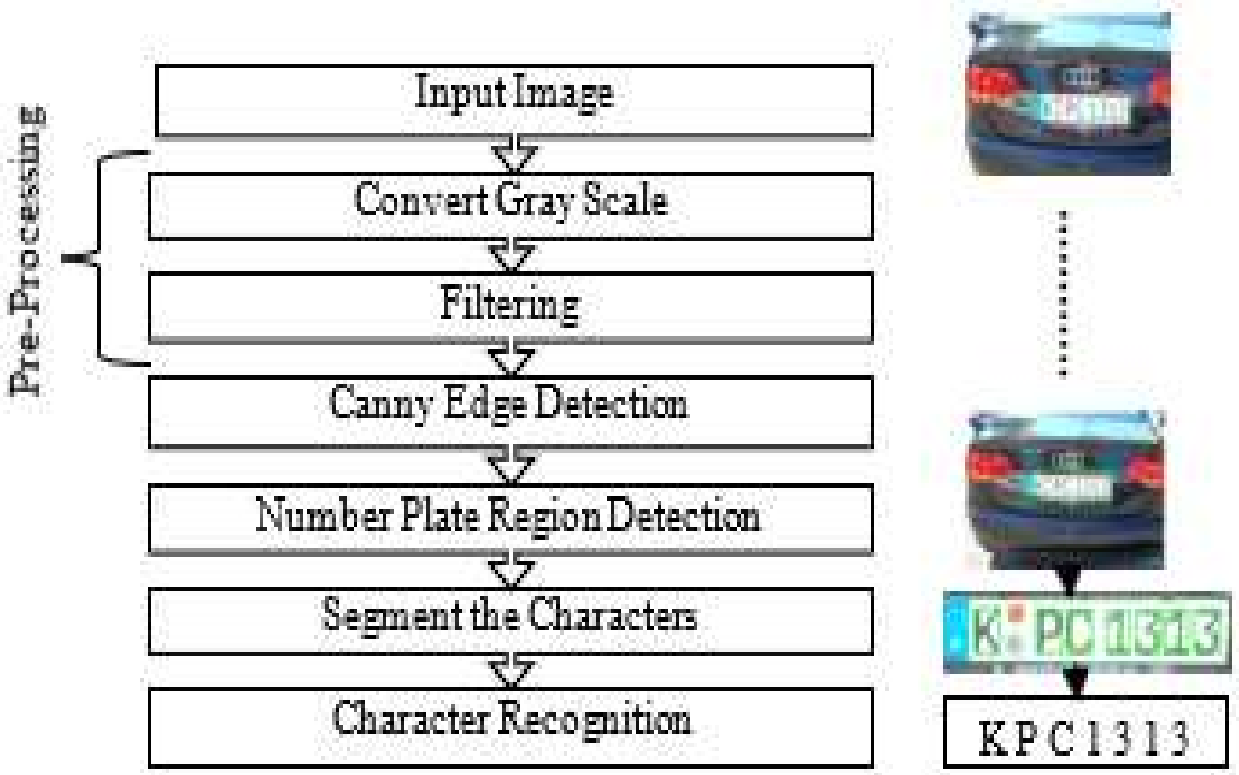


Fig – 3.1: Process Flow

3.2 Title:

Department of IT, Volume: 05 Issue: 03 | Mar-2018, "VEHICLE NUMBER PLATE DETECTION USING IMAGE PROCESSING," Abhay Singh, Anand Kumar Gupta, Anmol Singh, Anuj Gupta, and Sherish Johri

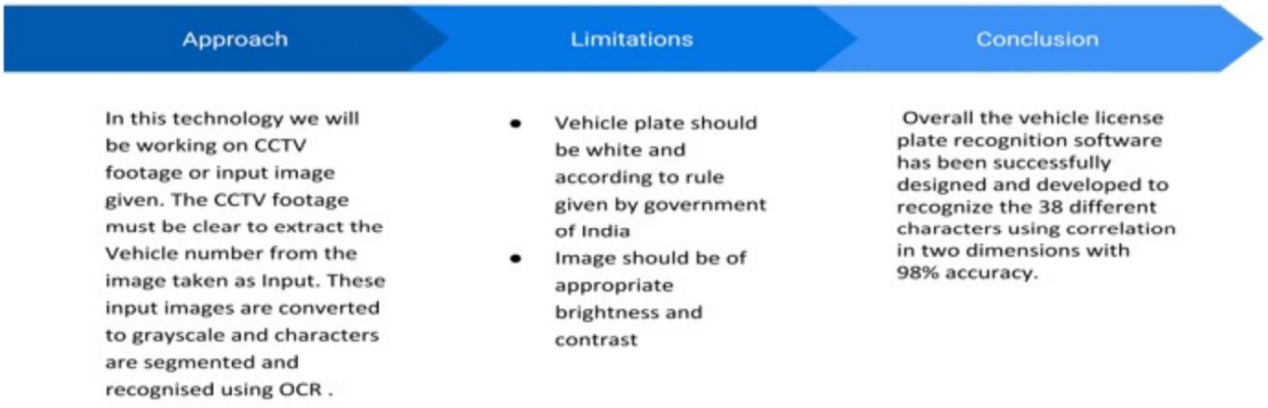
Context:

With this technique, we will be processing input photos or CCTV footage. To extract the vehicle number from the image used as input, the CCTV footage must be crystal clear. OCR is used to segment and recognize the characters in these input images once they have been transformed to grayscale. This software must meet a few requirements in order to function:

1) Automobile license plates must be white and in accordance with Indian government regulations.

2) Picture should have suitable contrast and brightness: This uses MATLAB to create software that recognizes the car number plate number.

With this procedure, we'll go through a number of methods step by step to discover the car number. Finally, using the discovered car number, we will compare that number from our database.



3.3 Title:

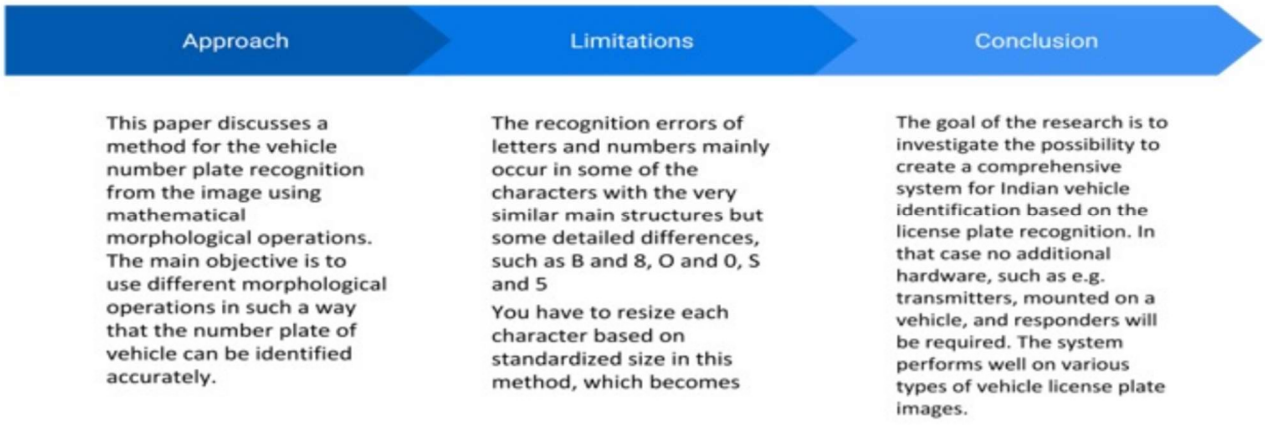
Ganesh R. Jadhav, Kailash J. Karande, "Automatic Vehicle Number Plate Recognition for Vehicle Parking Management System", IISTE, Vol.5, No.11, 2014.

Context:

The method for reading a vehicle's license plate from an image using mathematical morphological processes is covered in this work. The major goal is to accurately identify vehicle number plates by utilizing a variety of morphological techniques.

This is based on a number of techniques, including picture augmentation, morphological transformation, edge detection, and number plate extraction from vehicle photos. Following this segmentation, template matching is used to identify the characters on the license plate. This programme can quickly and precisely identify the number plate from the image of the car. The study's objective is to determine whether it is feasible to develop an extensive system for Indian vehicle identification based on license plate recognition. In that case, no additional hardware, like responders and transmitters installed on vehicles, will be necessary. The technology works effectively with a variety of photos of automobile license plates.

The characters with extremely similar fundamental structures but subtle changes, such B and 8, O and 0, and S and 5, tend to have letter and number identification issues. In this approach, each character must be resized in accordance with standardized size, adding a pre-processing step and lengthening the processing time.



4. PROBLEM IDENTIFICATION AND OBJECTIVES

4.1 Problem Identification:

The problem of number plate detection using a web camera is a computer vision task that involves developing an algorithm that can identify and extract the number plates of vehicles from a video stream captured by a webcam. This task can be useful in a variety of circumstances, including traffic control, parking regulation, and law enforcement. The main challenge in this task is to accurately identify and extract the number plates from the video stream, despite variations in lighting, vehicle orientation, and plate size.

4.2 Existing System:

4.2.1 Online ANPR framework: Using an online ALPR framework, tags are quickly limited and clarified from approaching video outlines, allowing for real-time tracking using the security camera.
Example: OpenALPR CloudWatch

4.2.2 Offline ANPR framework: Interestingly, a logged-off ALPR framework captures the shovel and dumper number plate images and stores them in a concentrated information server for later processing, i.e. for vehicle number plate translation.
Example: OpenALPR Library

4.3 Drawback:

Cities like Bangalore have several apartment complexes and businesses, most of which you can also verify by looking at the membership sticker on the vehicle's windshield. If a stranger or an unfamiliar car comes in, it has to be registered, which takes time. Most complexes are even considered unsafe once a vehicle enters, as it is difficult to track the movement of vehicle members. Security issues are a major drawback when many cars are stolen, especially when parked in parking lots. Keeping a record of all the vehicles entering and exiting during peak hours is difficult for several hours. With these shortcomings of traditional systems in mind, we want to go one step further while building our solutions and approaching each one individually.

4.4 Proposed system:

Automated licence plate detection using the huge and robust image processing library of OpenCV and an effective OCR engine like Py Tesseract. As we have seen, the majority of the problems we have identified are addressed by the ANPR cover. I'd like to now go a bit farther and outline the project's overall scope as well as the potential limits. Noise introduced to the image during image capture or from the environment is often the main issue in licence plate recognition. We may claim that our system can be used in any setting. either rain or night. A potential customer's major concern when a new system is suggested is typically how to add new features to the existing system. Considering this, we are sure that our system can be integrated into most of our customers' existing infrastructures. Using a web crawler, recognized license plates are parsed and sent to his government website, vahan.nic.in, along with the resolved captcha. Vehicle details can be retrieved for further inference and analysis. It also highlights the government's security weaknesses on his website and the government's privacy issues on his website. We also provide analysis and solutions for the extracted data.

Objectives:

The main goal of the project is to create a system that can precisely identify and extract licence plates from a web camera's video stream. The following are some specific objectives that can help achieve this goal:

1) Develop an algorithm that can accurately detect the number plates in a video stream, even in challenging conditions such as low light or occlusion.
2) Implement Using image processing methods like edge detection and segmentation, the number plate may be precisely extracted from the image.
3) Develop a user-friendly interface that can display the captured video stream and the extracted number plates.
4) Optimize the algorithm for real-time processing to guarantee the system's functionality in real-world situations.
5) Use relevant measures, such as accuracy, speed, and resilience, to assess the system's performance.

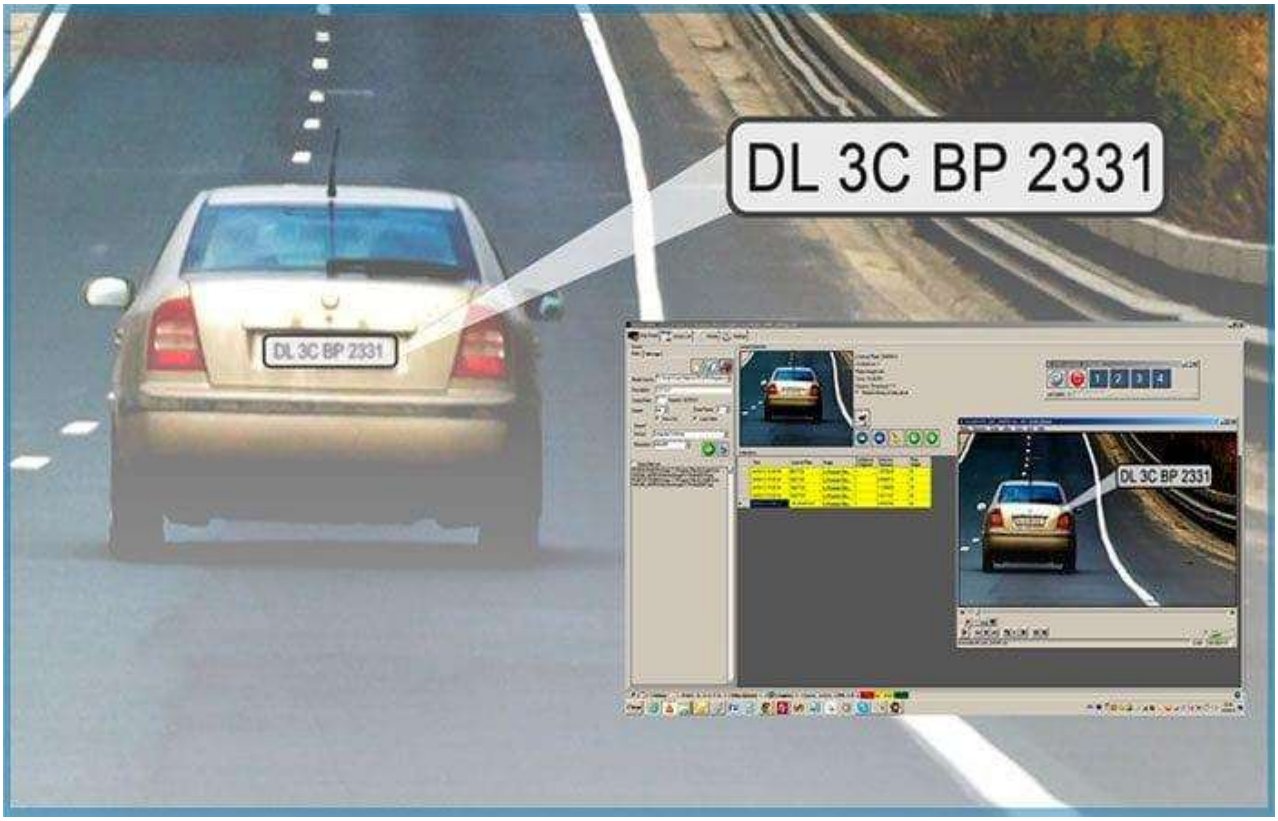


Fig – 4.1: OpenCV Extraction

5. SYSTEM METHODOLOGY

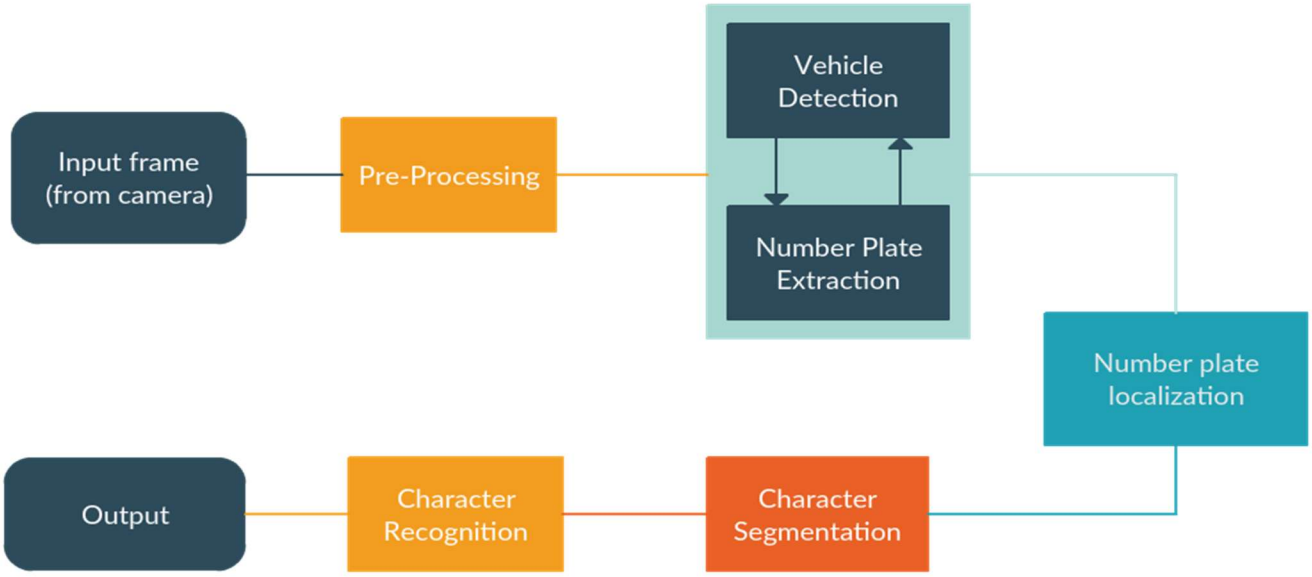


Fig 5.1: Framework

To implement a system for number plate detection using a web camera, we follow these general steps:

Data collection: Gather a dataset of images of vehicles with number plates. You can use publicly available datasets or capture your own images.

Preprocessing: Preprocess the images to remove noise, adjust brightness and contrast, and apply image enhancement techniques if necessary.

Object detection: Use an object detection algorithm, such as Haar Cascades, to identify and localize the regions in the image that contain number plates.

Number plate extraction: Use image segmentation and cropping techniques to isolate the number plate area from the surrounding background.

Character segmentation: Segment the characters on the licence plate using linked component analysis or other techniques.

Characters recognition': Recognize the characters usage of optical character recognition (OCR) algorithms.

Output: Display the recognized number plate on the screen or store it in a database for further processing.

6. OVERVIEW OF TECHNOLOGIES

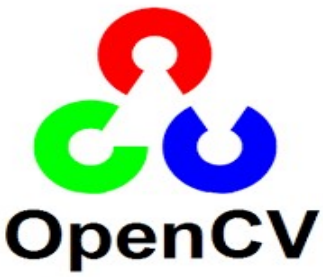
6.1 OPEN CV

The acronym for OpenCV is "Open Source Computer Vision Library." It is an open-source software library for computer vision and machine learning that is free and aimed at assisting programmers in creating real-time computer vision applications. For the processing of images and videos, object detection and recognition, and other tasks, OpenCV provides a variety of tools and features. It is written in the C++ programming language and supports Python, Java, and MATLAB among other computer languages. OpenCV is widely used in various industries, including robotics, surveillance, automotive, healthcare, and more. Its vast range of features and functionality has made it one of the most popular computer vision libraries in the world.

1.0 was the initial version of OpenCV. Both academic and commercial use of OpenCV are free because it is made available under a BSD licence. It offers interfaces in C++, C, Python, and Java and supports Windows, Linux, Mac OS, iOS, and Android. When OpenCV was developed, real-time applications for effective processing were the primary focus. Everything has been optimised for multi-core processing and is written in C/C++.

6.1.1 Functions of OpenCV:-

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



6.1.2 Advantages of OpenCV:

A strong and adaptable tool for computer vision applications is OpenCV -Open -Source- Computer Vision- Library. Some of its main benefits are:

1) **Open-source and free:** OpenCV is a library that is free to use since it is open-source and can be used by anyone. This makes it accessible to developers of all skill levels.

2) **Cross-platform:** OpenCV supports Windows, Linux, macOS, Android, and iOS are just a few of the platforms available.. This means that developers can build applications for a variety of hardware and operating systems.

3) **Extensive library:** OpenCV offers a range of tools and functions for image and video processing, object detection and recognition, feature extraction, and more. This makes it easy for developers to implement complex computer vision algorithms without having to write everything from scratch.

4) **Active community:** OpenCV has a sizable and vibrant developer community that actively participates in the library's development. This ensures that the library stays up-to-date and offers the latest features.

5) **Multiple language support:** Python', C++', Java', and MATLAB' are just a few of the programming languages that OpenCV is compatible with. This makes it easy for developers to use the language they are most comfortable with.

6) **Integration with other libraries:** OpenCV can be easily integrated with other libraries such as TensorFlow, Keras, and PyTorch. This makes it possible to build complex computer vision applications that incorporate machine learning.

Overall, OpenCV is a powerful and versatile tool for computer vision applications that offers a range of advantages to developers.

6.1.3 OpenCV Disadvantages:

Popular open-source framework for computer vision and machine learning, OpenCV offers a wide range of features and capabilities for processing images and videos. While it offers many advantages, there are also some potential disadvantages of using OpenCV, which include:

1) **Steep Learning Curve:** OpenCV is a complex library with many features, which can make it challenging for beginners to get started with. It requires a good understanding of programming concepts and computer vision algorithms.

2) **Performance:** Although OpenCV is optimized for performance, it can still be resource-intensive, especially when dealing with large datasets or real-time applications. This can result in slower processing times and higher hardware requirements.

3) **Compatibility:** OpenCV supports a variety of programming languages, such as Java', Python', and C + +, but not all functions and features are available in all languages. This can make it difficult to work with OpenCV in some programming environments.

4) **Limited Documentation:** While there is a wealth of information available online about OpenCV, the official documentation can be limited in some areas, especially when it comes to more advanced topics or newer features.

5) **Lack of Machine Learning Models:** While OpenCV has a range of computer vision algorithms, it does not have many pre-trained machine learning models. This can make it more challenging to implement advanced machine learning techniques without additional programming.

6) **Limited Support for 3D Vision:** OpenCV is primarily designed for 2D image and video processing, and while it does have some support for 3D vision, it is not as comprehensive as other libraries designed specifically for 3D computer vision

6.2 OCR

OCR stands for Character recognition using optical. With this technique, computers can recognize printed or handwritten text in digital images, scanned documents, or photographs, and convert it into editable and searchable electronic text.

OCR technology uses complex algorithms and machine learning models to analyze the shapes, patterns, and colors of characters in an image and match them with a database of known characters. The OCR software then converts the image into an editable and searchable text file.

OCR technology is commonly used in many industries, including finance, healthcare, legal, and education, to digitize paper documents and streamline document processing workflows. It It may be utilized for projects like digitizing invoices, forms, receipts, and handwritten notes, among others..

OCR solutions increase users' access to information
OCR technology is frequently used to automatically convert image-based files like PDFs, TIFFs, and JPGs into text-based, machine-readable files. Receipts, contracts, invoices, financial statements, and other documents that have undergone OCR processing can be:

- Searched from a sizable repository to discover the right document
- Viewed with the ability to search within each document.
- Edited to make necessary corrections.
- Repurposed by having the extracted text sent to other systems.

How does optical character recognition work?

Optical Character Recognition (OCR) works by analyzing the shapes, patterns, and colors of characters in an image and converting them into machine-readable text. Here are the basic steps involved in OCR:
Image capture: The OCR process begins with the capture of an image, which can be a scanned document, a photograph, or a digital image.

Pre-processing: The image is pre-processed to remove any noise, distortion, or background interference

that might affect character recognition's precision.

Characters segmentation': The image is then analyzed to identify individual characters, which are separated from each other and segmented into individual images.

Feature extraction: The segmented characters are analyzed to extract their unique features, such as size, shape, and color.

Character recognition: The extracted features are compared to a database of known characters to identify the most likely match for each character.

Post-processing: The recognized characters are processed to correct any errors or ambiguities, and the final output is generated as machine-readable text.

OCR technology uses a combination of algorithms and machine learning models to perform these steps and achieve high accuracy rates in character recognition. The quality of the original image, the complexity of the characters, and the language used can all affect the accuracy of OCR, so it's important to ensure that the input is of good quality and that the OCR software is properly trained and optimized for the task at hand.

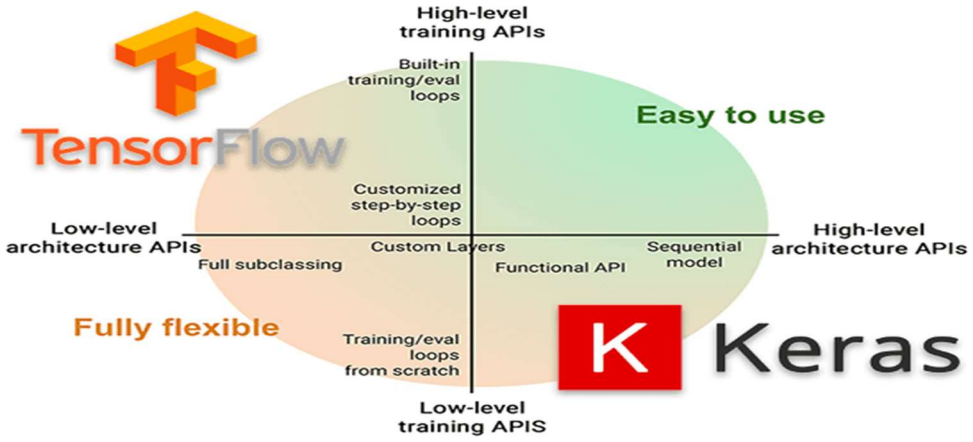
6.3 TensorFlow and Keras

TensorFlow and Keras are two popular deep learning frameworks used for building and training neural systems.

Data flow graphs are used in TensorFlow, an open-source software library for numerical computing. It was developed by the Google Brain team and released in 2015. Building and refining machine learning models, including neural networks, is made flexible by TensorFlow. It is compatible with a wide range of systems, including Central Processing Unit , Graphics Processing Unit and Tensor Processing Unit , and supports a number of programming languages, including Python', C'++, and Java'.

Keras is an advanced API constructed on top of TensorFlow. (and other deep learning frameworks) For image classification, object identification, and computer vision applications, CNNs are a type of neural networks that is often employed in deep learning. Keras was developed by François Chollet and first released in 2015. It provides a user-friendly interface for building and training deep learning models, with a focus on ease of use, modularity, and extensibility. Keras supports multiple backends, including TensorFlow, Theano, and CNTK, allowing users to switch between different frameworks without having to change their code.

While TensorFlow is a more low-level framework that provides more control and flexibility, Keras offers a higher-level API that makes it easier to build and train models quickly. Many deep learning practitioners use a combination of both TensorFlow and Keras, leveraging the strengths of each framework for different tasks.



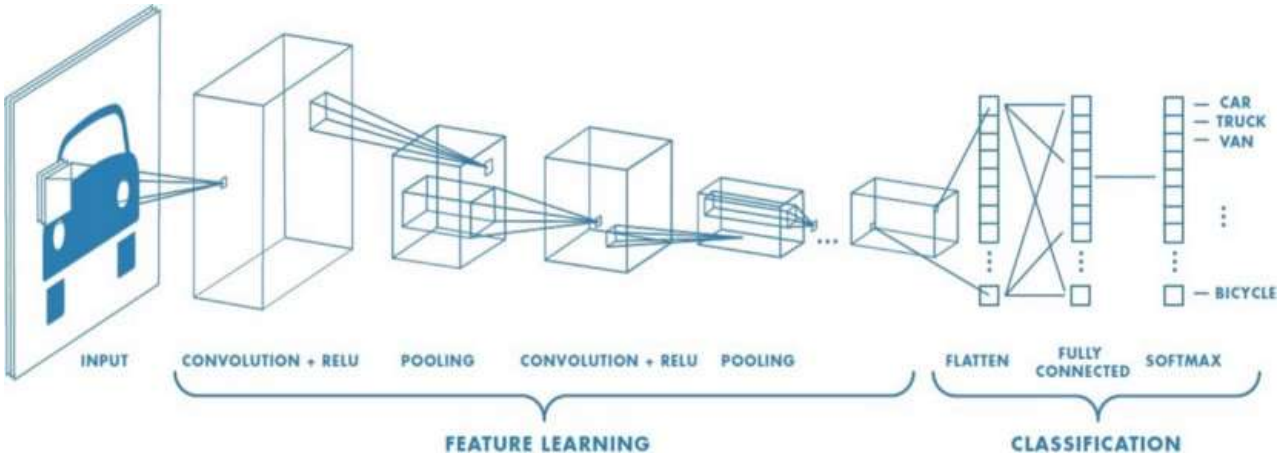
6.4 CNN

A convolutional-neural-networks, a type of neural network (CNNs) is often used in deep learning for computer vision, object identification, and picture categorization applications.
The structure of the visual cortex in animals, where individual neurons respond to certain areas of the visual field, served as the model for CNNs'. Similar to this, a CNN' is made up of a number of layers that recognise and extract features from images.

A CNN, the fundamental unit of a convolutional-layers, applies a number of filters. (sometimes referred to as kernels or weights) to an input picture to generate a number of feature maps as the result. Each filter pulls a particular feature from the input image, such as edges, corners, or forms. To lower the spatial dimensions and improve the non-linearities of the features, extra layers can be added to the output feature maps, such as pooling and activation layers.

CNNs can be trained using backpropagation, a supervised learning algorithm that adjusts the filter weights to reduce the discrepancy between the output labels expected and actual. In order to teach the network the necessary features for each class, a huge collection of labelled photos are fed into it, and the weights of the filters are changed.
CNNs have been successful a variety of applications, including face recognition, object identification,

and medical image analysis, and are considered a state-of-the-art technique in computer vision.



6.4.1 Advantages of CNN:

Convolutional Neural Networks (CNNs) have several advantages over other types of neural networks for image and video processing tasks. Here are some of the key advantages:

1) **Effective feature extraction:** CNNs can automatically learn high-level features from raw image pixels without the need for manual feature engineering. This is because CNNs use a series of convolutional layers that detect and extract features at different spatial scales, making them well-suited for the purposes of object identification, picture categorization, and other computer vision applications.

2) **Translation invariance:** CNNs are able to recognize objects even if an item is moved in a picture or rotated relative to its original position. This is so that features can be detected regardless of their spatial placement within the image, as convolutional-layers are created to be translation invariant.
3) **Parameter sharing:** CNN's use parameter sharing to reduce the number of parameters needed to train the network. This is because Many areas in the input image are subjected to the same filter,allowing the network to learn a smaller set of parameters that can be reused across the entire image.

4) **Data efficiency:** CNNs are able to learn from relatively small datasets compared to other types of neural networks. This is because the convolutional layers act as feature extractors that can generalize across different image sizes and orientations, making the network more robust to variations in the data.

5) **State-of-the-art performance:** In numerous computer vision tasks, such as picture classification, object recognition, and semantic segmentation, CNN's have reached state-of-the-art performance. This is because CNN's are able to capture complex, non-linear relationships between features in the input image, making them well-suited for deep learning applications.

6.4.1 Disadvantages of CNN:

Despite the many benefits of convolutional neural networks (CNNs), there are some possible drawbacks of employing

1) **Computational complexity:** CNNs can be computationally expensive to train and require a lot of memory and processing power, especially for large datasets and deep architectures. This can make training and testing slow and expensive, and may require specialized hardware, such as GPUs or TPUs.

2) **Overfitting:** In the same way as other deep learning models, CNNs are vulnerable the network becomes highly specialized to the training data and underperforms on new, uncontaminated data due to overfitting.. Dropout and weight decay are two regularization methods that can reduce overfitting but can increase the model's complexity and training time.

3) **Limited interpretability:** While CNNs are able to learn complex representations of the input data, it can be difficult to interpret how the network is making its predictions. This can be problematic in applications where transparency and interpretability are important, such as medical diagnosis or legal decision-making.

4) **Sensitivity to hyperparameters:** CNNs have many hyperparameters, such as the number of layers, the size of the filters, and the learning rate, that need to be tuned carefully to achieve optimal performance. This can be a time-consuming and iterative process that requires a lot of experimentation and computational resources.

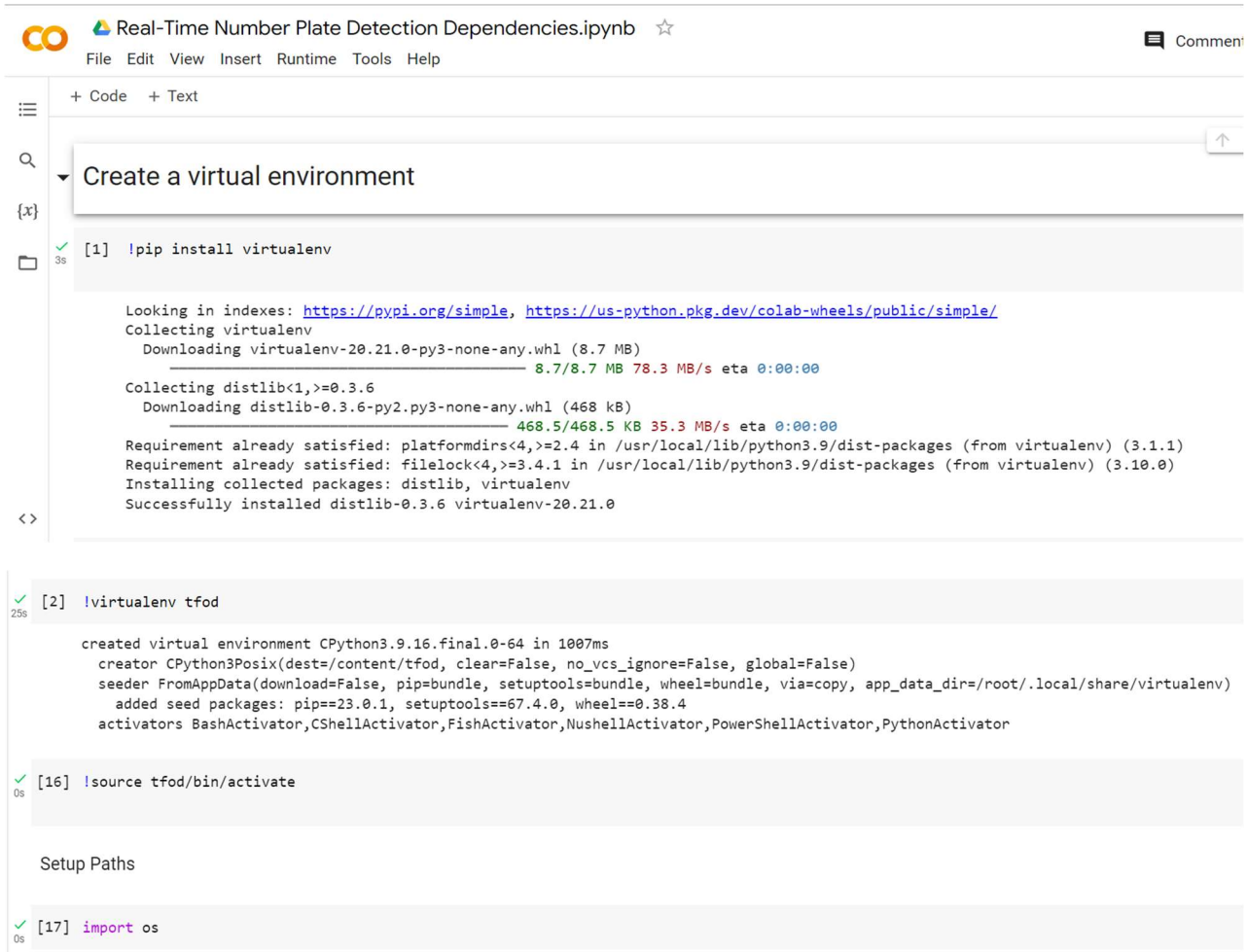
5) **Requires large amounts of labeled data:** CNNs require a lot of labelled training data, which might be challenging in some domains where annotated training data is expensive or hard to come by This can limit the applicability of CNNs to certain tasks and domains.

7. IMPLEMENTATION':

Fig 7.1: Final Output

All of the models in the architecture are implemented through Python programming, utilising the necessary tools.

Coding and Testing



```
[7]  files = {
         'PIPELINE_CONFIG':os.path.join('Tensorflow', 'workspace','models', CUSTOM_MODEL_NAME, 'pipeline.config'),
         'TF_RECORD_SCRIPT': os.path.join(paths['SCRIPTS_PATH'], TF_RECORD_SCRIPT_NAME),
         'LABELMAP': os.path.join(paths['ANNOTATION_PATH'], LABEL_MAP_NAME)
     }
```

```
for path in paths.values():
    if not os.path.exists(path):
        if os.name == 'posix':
            !mkdir -p {path}
        if os.name == 'nt':
            !mkdir {path}
```

1. Download TF Models Pretrained Models from Tensorflow Model Zoo and Install TFOD

```
[9]  if os.name=='nt':
         !pip install wget
         import wget
```

```
[10] if not os.path.exists(os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection')):
         !git clone https://github.com/tensorflow/models {paths['APIMODEL_PATH']}
```

```
Cloning into 'Tensorflow/models'...
remote: Enumerating objects: 82517, done.
remote: Counting objects: 100% (423/423), done.
remote: Compressing objects: 100% (187/187), done.
remote: Total 82517 (delta 265), reused 372 (delta 236), pack-reused 82094
Receiving objects: 100% (82517/82517), 596.52 MiB | 26.05 MiB/s, done.
Resolving deltas: 100% (58888/58888), done.
```

```
[11] # Install Tensorflow Object Detection
     if os.name=='posix':
         !apt-get install protobuf-compiler
         !cd Tensorflow/models/research && protobuf object_detection/protos/*.proto --python_out=. && cp object_detection/packages/tf2/setup.py . && python
     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'
```

```
if os.name=='nt':
    url="https://github.com/protocolbuffers/protobuf/releases/download/v3.15.6/protoc-3.15.6-win64.zip"
    wget.download(url)
    !move protoc-3.15.6-win64.zip {paths['PROTOC_PATH']}
    !cd {paths['PROTOC_PATH']} && tar -xf protoc-3.15.6-win64.zip
    os.environ['PATH'] += os.pathsep + os.path.abspath(os.path.join(paths['PROTOC_PATH'], 'bin'))
    !cd Tensorflow/models/research && protobuf object_detection/protos/*.proto --python_out=. && copy object_detection\\packages\\tf2\\setup.py setup.py
    !cd Tensorflow/models/research/slim && pip install -e .
```

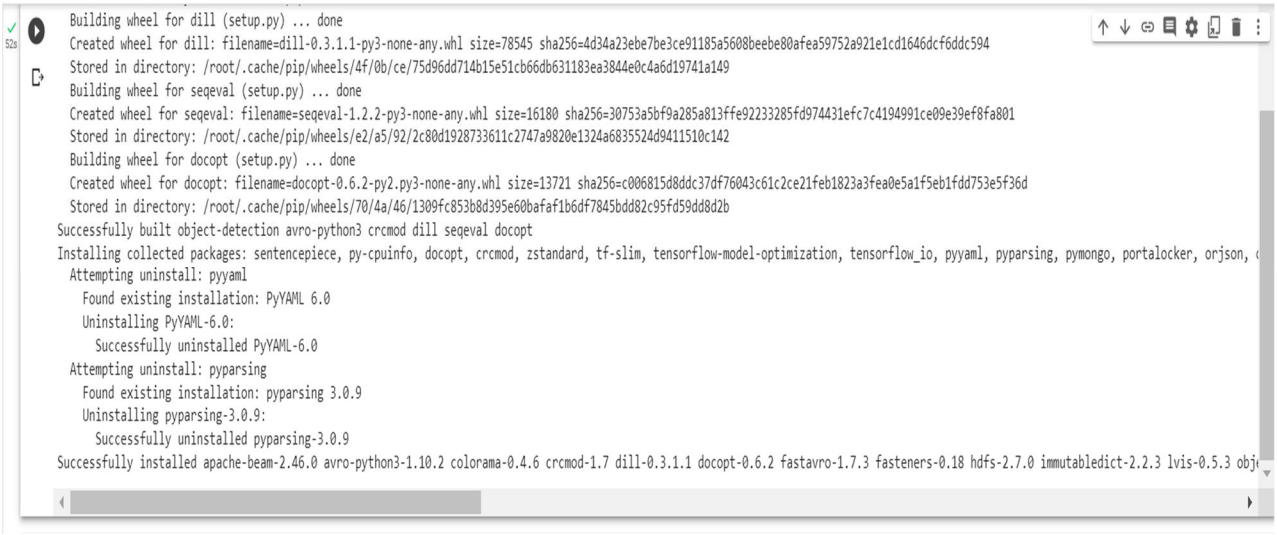
```
Reading package lists... Done
Building dependency tree
Reading state information... Done
protobuf-compiler is already the newest version (3.6.1.3-2ubuntu5.2).
0 upgraded, 0 newly installed, 0 to remove and 23 not upgraded.
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Processing /content/Tensorflow/models/research
  Preparing metadata (setup.py) ... done
Collecting avro-python3
  Downloading avro-python3-1.10.2.tar.gz (38 kB)
  Preparing metadata (setup.py) ... done
Collecting apache-beam
  Downloading apache_beam-2.46.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (14.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14.5/14.5 MB 48.0 MB/s eta 0:00:00
Requirement already satisfied: pillow in /usr/local/lib/python3.9/dist-packages (from object-detection==0.1) (8.4.0)
Requirement already satisfied: lxml in /usr/local/lib/python3.9/dist-packages (from object-detection==0.1) (4.9.2)
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.9/dist-packages (from object-detection==0.1) (3.7.1)
Requirement already satisfied: Cython in /usr/local/lib/python3.9/dist-packages (from object-detection==0.1) (0.29.33)
Requirement already satisfied: contextlib2 in /usr/local/lib/python3.9/dist-packages (from object-detection==0.1) (0.6.0.post1)
Collecting tf-slim
  Downloading tf_slim-1.1.0-py2.py3-none-any.whl (352 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 352.1/352.1 KB 25.8 MB/s eta 0:00:00
Requirement already satisfied: six in /usr/local/lib/python3.9/dist-packages (from object-detection==0.1) (1.16.0)
Requirement already satisfied: pycocotools in /usr/local/lib/python3.9/dist-packages (from object-detection==0.1) (2.0.6)
Collecting lvis
  Downloading lvis-0.5.3-py3-none-any.whl (14 kB)
Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-packages (from object-detection==0.1) (1.10.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages (from object-detection==0.1) (1.4.4)
Collecting tf-models-official>=2.5.1
  Downloading tf_models_official-2.11.4-py2.py3-none-any.whl (2.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.4/2.4 MB 13.3 MB/s eta 0:00:00
Collecting tensorflow_io
  Downloading tensorflow_io-0.31.0-cp39-cp39-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (26.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 26.9/26.9 MB 13.4 MB/s eta 0:00:00
Requirement already satisfied: keras in /usr/local/lib/python3.9/dist-packages (from object-detection==0.1) (2.11.0)
Collecting pyparsing==2.4.7
  Downloading pyparsing-2.4.7-py2.py3-none-any.whl (67 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 67.8/67.8 KB 4.1 MB/s eta 0:00:00
Collecting sacrebleu<=2.2.0
  Downloading sacrebleu-2.2.0-py3-none-any.whl (116 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.6/116.6 KB 9.9 MB/s eta 0:00:00
```

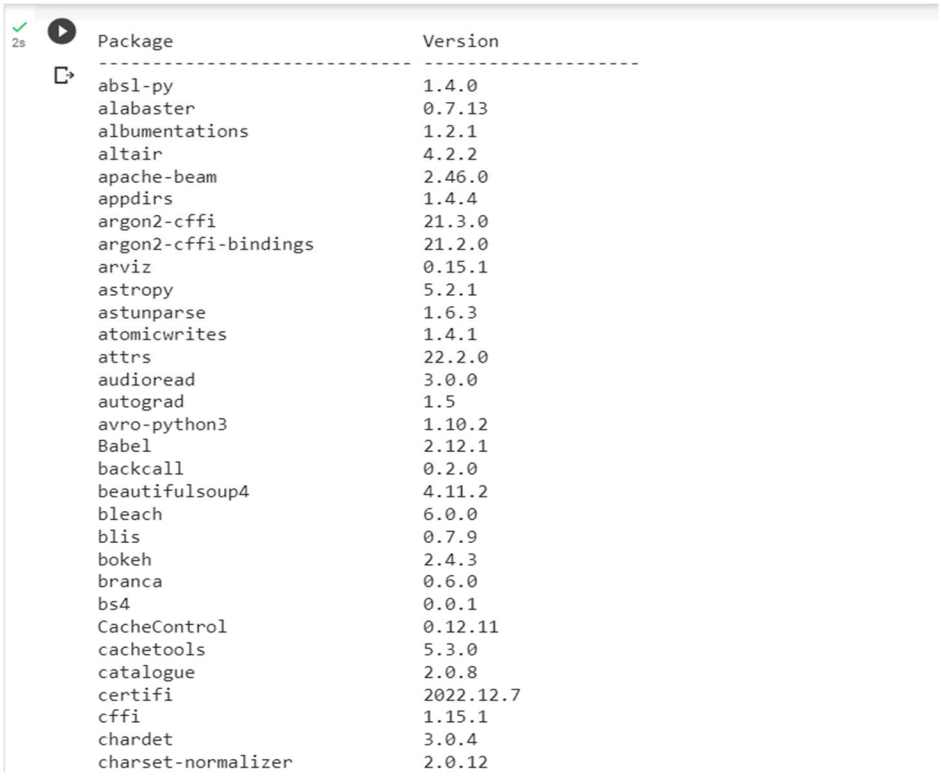
```
Requirement already satisfied: regex in /usr/local/lib/python3.9/dist-packages (from sacrebleu<=2.2.0->object-detection==0.1) (2022.10.31)
Collecting colorama
  Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.9/dist-packages (from sacrebleu<=2.2.0->object-detection==0.1) (1.22.4)
Requirement already satisfied: tabulate>=0.8.9 in /usr/local/lib/python3.9/dist-packages (from sacrebleu<=2.2.0->object-detection==0.1) (0.8.10)
Collecting portalocker
  Downloading portalocker-2.7.0-py2.py3-none-any.whl (15 kB)
Collecting sentencepiece
  Downloading sentencepiece-0.1.97-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 46.3 MB/s eta 0:00:00
Requirement already satisfied: kaggle>=1.3.9 in /usr/local/lib/python3.9/dist-packages (from tf-models-official>=2.5.1->object-detection==0.1) (1.5.
Collecting immutabledict
  Downloading immutabledict-2.2.3-py3-none-any.whl (4.0 kB)
Requirement already satisfied: gin-config in /usr/local/lib/python3.9/dist-packages (from tf-models-official>=2.5.1->object-detection==0.1) (0.5.0)
Collecting tensorflow-model-optimization>=0.4.1
  Downloading tensorflow_model_optimization-0.7.3-py2.py3-none-any.whl (238 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 238.9/238.9 KB 18.2 MB/s eta 0:00:00
Requirement already satisfied: tensorflow-datasets in /usr/local/lib/python3.9/dist-packages (from tf-models-official>=2.5.1->object-detection==0.1)
Requirement already satisfied: google-api-python-client>=1.6.7 in /usr/local/lib/python3.9/dist-packages (from tf-models-official>=2.5.1->object-det
Collecting seqeval
  Downloading seqeval-1.2.2.tar.gz (43 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 43.6/43.6 KB 4.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: tensorflow-hub>=0.6.0 in /usr/local/lib/python3.9/dist-packages (from tf-models-official>=2.5.1->object-detection==0.
Requirement already satisfied: tensorflow~=2.11.0 in /usr/local/lib/python3.9/dist-packages (from tf-models-official>=2.5.1->object-detection==0.1)
```

```
  Preparing metadata (setup.py) ... done
Requirement already satisfied: tensorflow-hub>=0.6.0 in /usr/local/lib/python3.9/dist-packages (from tf-models-official>=2.5.1->object-detection==0.1) (0.13.0)
Requirement already satisfied: tensorflow~=2.11.0 in /usr/local/lib/python3.9/dist-packages (from tf-models-official>=2.5.1->object-detection==0.1) (2.11.0)
Requirement already satisfied: oauth2client in /usr/local/lib/python3.9/dist-packages (from tf-models-official>=2.5.1->object-detection==0.1) (4.1.3)
Requirement already satisfied: opencv-python-headless in /usr/local/lib/python3.9/dist-packages (from tf-models-official>=2.5.1->object-detection==0.1) (4.7.0.72)
Collecting tensorflow-text~=2.11.0
  Downloading tensorflow_text-2.11.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (5.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.8/5.8 MB 87.0 MB/s eta 0:00:00
Collecting tensorflow-addons
  Downloading tensorflow_addons-0.19.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 63.9 MB/s eta 0:00:00
Collecting pyyaml<6.0,>=5.1
  Downloading PyYAML-5.4.1-cp39-cp39-manylinux1_x86_64.whl (630 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 630.1/630.1 KB 46.2 MB/s eta 0:00:00
Collecting py-cpuinfo>=3.3.0
  Downloading py_cpuinfo-9.0.0-py3-none-any.whl (22 kB)
Requirement already satisfied: psutil>=5.4.3 in /usr/local/lib/python3.9/dist-packages (from tf-models-official>=2.5.1->object-detection==0.1) (5.9.4)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas->object-detection==0.1) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas->object-detection==0.1) (2.8.2)
Requirement already satisfied: absl-py>=0.2.2 in /usr/local/lib/python3.9/dist-packages (from tf-slim->object-detection==0.1) (1.4.0)
Collecting orjson<4.0
  Downloading orjson-3.8.8-cp39-cp39-manylinux_2_28_x86_64.whl (143 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 143.5/143.5 KB 15.3 MB/s eta 0:00:00
Requirement already satisfied: pydot<2,>=1.2.0 in /usr/local/lib/python3.9/dist-packages (from apache-beam->object-detection==0.1) (1.4.2)
Requirement already satisfied: pyarrow<10.0.0,>=3.0.0 in /usr/local/lib/python3.9/dist-packages (from apache-beam->object-detection==0.1) (9.0.0)
Collecting fasteners<1.0,>=0.3
  Downloading fasteners-0.18-py3-none-any.whl (18 kB)
Requirement already satisfied: requests<3.0.0,>=2.24.0 in /usr/local/lib/python3.9/dist-packages (from apache-beam->object-detection==0.1) (2.27.1)
Requirement already satisfied: proto-plus<2,>=1.7.1 in /usr/local/lib/python3.9/dist-packages (from apache-beam->object-detection==0.1) (1.22.2)
Collecting zstandard<1,>=0.18.0
  Downloading zstandard-0.20.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.6/2.6 MB 69.1 MB/s eta 0:00:00
Collecting hdfs<3.0.0,>=2.1.0
  Downloading hdfs-2.7.0-py3-none-any.whl (34 kB)
```

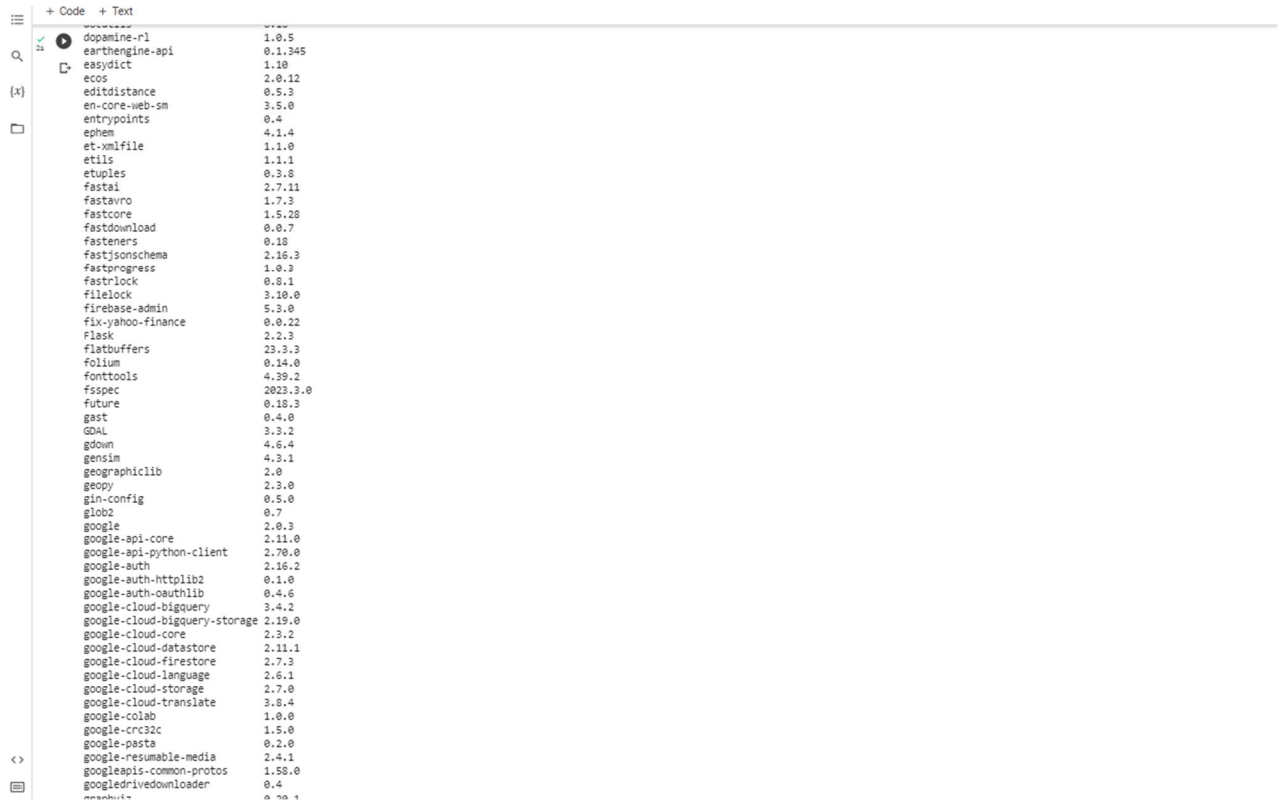
```
  Downloading hdfs-2.7.0-py3-none-any.whl (34 kB)
Collecting obsize<0.7.0,>=0.6.1
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->object-detection==0.1) (23.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->object-detection==0.1) (4.39.2)
Requirement already satisfied: importlib-resources>=3.2.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->object-detection==0.1) (5.12.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib->object-detection==0.1) (1.0.7)
Requirement already satisfied: tensorflow-io-gcs-filesystem==0.31.0 in /usr/local/lib/python3.9/dist-packages (from tensorflow_io->object-detection==0.1) (0.31.0)
Requirement already satisfied: google-auth<3.0.0dev,>=1.19.0 in /usr/local/lib/python3.9/dist-packages (from google-api-python-client>=1.6.7->tf-models-official>=2.5.1->object-detection==0.1
Requirement already satisfied: google-auth-httplib2>=0.1.0 in /usr/local/lib/python3.9/dist-packages (from google-api-python-client>=1.6.7->tf-models-official>=2.5.1->object-detection==0.1)
Requirement already satisfied: uritemplate<5,>=3.0.1 in /usr/local/lib/python3.9/dist-packages (from google-api-python-client>=1.6.7->tf-models-official>=2.5.1->object-detection==0.1) (4.1.1
Requirement already satisfied: google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5 in /usr/local/lib/python3.9/dist-packages (from google-api-python-client>=1.6.7->tf-models-o
Collecting docopt
  Downloading docopt-0.6.2.tar.gz (25 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.9/dist-packages (from importlib-resources>=3.2.0->matplotlib->object-detection==0.1) (3.15.0)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.9/dist-packages (from kaggle>=1.3.9->tf-models-official>=2.5.1->object-detection==0.1) (1.26.15)
Requirement already satisfied: certifi in /usr/local/lib/python3.9/dist-packages (from kaggle>=1.3.9->tf-models-official>=2.5.1->object-detection==0.1) (2022.12.7)
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-packages (from kaggle>=1.3.9->tf-models-official>=2.5.1->object-detection==0.1) (4.65.0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.9/dist-packages (from kaggle>=1.3.9->tf-models-official>=2.5.1->object-detection==0.1) (8.0.1)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests<3.0.0,>=2.24.0->apache-beam->object-detection==0.1) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests<3.0.0,>=2.24.0->apache-beam->object-detection==0.1) (3.4)
Requirement already satisfied: setuptools in /usr/local/lib/python3.9/dist-packages (from tensorflow~=2.11.0->tf-models-official>=2.5.1->object-detection==0.1) (67.6.0)
Requirement already satisfied: flatbuffers>=2.0 in /usr/local/lib/python3.9/dist-packages (from tensorflow~=2.11.0->tf-models-official>=2.5.1->object-detection==0.1) (23.3.3)
Requirement already satisfied: tensorflow-estimator<2.12,>=2.11.0 in /usr/local/lib/python3.9/dist-packages (from tensorflow~=2.11.0->tf-models-official>=2.5.1->object-detection==0.1) (2.11
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.9/dist-packages (from tensorflow~=2.11.0->tf-models-official>=2.5.1->object-detection==0.1) (3.3.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.9/dist-packages (from tensorflow~=2.11.0->tf-models-official>=2.5.1->object-detection==0.1) (2.2.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.9/dist-packages (from tensorflow~=2.11.0->tf-models-official>=2.5.1->object-detection==0.1) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.9/dist-packages (from tensorflow~=2.11.0->tf-models-official>=2.5.1->object-detection==0.1) (15.0.6.1)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.9/dist-packages (from tensorflow~=2.11.0->tf-models-official>=2.5.1->object-detection==0.1) (1.15.0)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /usr/local/lib/python3.9/dist-packages (from tensorflow~=2.11.0->tf-models-official>=2.5.1->object-detection==0.1) (0.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.9/dist-packages (from tensorflow~=2.11.0->tf-models-official>=2.5.1->object-detection==0.1) (1.6.3)
Requirement already satisfied: tensorboard<2.12,>=2.11 in /usr/local/lib/python3.9/dist-packages (from tensorflow~=2.11.0->tf-models-official>=2.5.1->object-detection==0.1) (2.11.2)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.9/dist-packages (from tensorflow~=2.11.0->tf-models-official>=2.5.1->object-detection==0.1) (3.8.0)
Requirement already satisfied: dm-tree~=0.1.1 in /usr/local/lib/python3.9/dist-packages (from tensorflow-model-optimization>=0.4.1->tf-models-official>=2.5.1->object-detection==0.1) (0.1.8)
Requirement already satisfied: pyasn1-modules>=0.0.5 in /usr/local/lib/python3.9/dist-packages (from oauth2client->tf-models-official>=2.5.1->object-detection==0.1) (0.2.8)
Requirement already satisfied: rsa>=3.1.4 in /usr/local/lib/python3.9/dist-packages (from oauth2client->tf-models-official>=2.5.1->object-detection==0.1) (4.9)
```


Building wheel for dill (setup.py) ... done
Created wheel for dill: filename=dill-0.3.1.1-py3-none-any.whl size=78545 sha256=4d34a23ebe7be3ce91185a5608beebe80afea59752a921e1cd1646dcf6ddc594
Stored in directory: /root/.cache/pip/wheels/4f/0b/ce/75d96dd714b15e51cb66db631183ea3844e0c4a6d19741a149
Building wheel for seqeval (setup.py) ... done
Created wheel for seqeval: filename=seqeval-1.2.2-py3-none-any.whl size=16180 sha256=30753a5bf9a285a813ffe92233285fd974431efc7c4194991ce09e39ef8fa801
Stored in directory: /root/.cache/pip/wheels/e2/a5/92/2c80d1928733611c2747a9820e1324a6835524d9411510c142
Building wheel for docopt (setup.py) ... done
Created wheel for docopt: filename=docopt-0.6.2-py2.py3-none-any.whl size=13721 sha256=c006815d8ddc37df76043c61c2ce21feb1823a3fea0e5a1f5eb1fdd753e5f36d
Stored in directory: /root/.cache/pip/wheels/70/4a/46/1309fc853b8d395e60bafaf1b6df7845bdd82c95fd59dd8d2b
Successfully built object-detection avro-python3 crcmod dill seqeval docopt
Installing collected packages: sentencepiece, py-cpuinfo, docopt, crcmod, zstandard, tf-slim, tensorflow-model-optimization, tensorflow_io, pyyaml, pyparsing, pymongo, portalocker, orjson,
Attempting uninstall: pyyaml
Found existing installation: PyYAML 6.0
Uninstalling PyYAML-6.0:
Successfully uninstalled PyYAML-6.0
Attempting uninstall: pyparsing
Found existing installation: pyparsing 3.0.9
Uninstalling pyparsing-3.0.9:
Successfully uninstalled pyparsing-3.0.9
Successfully installed apache-beam-2.46.0 avro-python3-1.10.2 colorama-0.4.6 crcmod-1.7 dill-0.3.1.1 docopt-0.6.2 fastavro-1.7.3 fasteners-0.18 hdfs-2.7.0 immutabledict-2.2.3 lvis-0.5.3


Requirement already satisfied: pyasn1>=0.1.7 in /usr/local/lib/python3.9/dist-packages (from oauth2client->tf-models-official>=2.5.1->object-detection==0.1) (0.4.8)
Requirement already satisfied: scikit-learn>=0.21.3 in /usr/local/lib/python3.9/dist-packages (from seqeval->tf-models-official>=2.5.1->object-detection==0.1) (1.2.2)
Collecting typeguard>=2.7
Downloading typeguard-3.0.2-py3-none-any.whl (30 kB)
Requirement already satisfied: etils[enp,epath]>=0.9.0 in /usr/local/lib/python3.9/dist-packages (from tensorflow-datasets->tf-models-official>=2.5.1->object-detection==0.1) (1.1.1)
Requirement already satisfied: tensorflow-metadata in /usr/local/lib/python3.9/dist-packages (from tensorflow-datasets->tf-models-official>=2.5.1->object-detection==0.1) (1.12.0)
Requirement already satisfied: promise in /usr/local/lib/python3.9/dist-packages (from tensorflow-datasets->tf-models-official>=2.5.1->object-detection==0.1) (2.3)
Requirement already satisfied: toml in /usr/local/lib/python3.9/dist-packages (from tensorflow-datasets->tf-models-official>=2.5.1->object-detection==0.1) (0.10.2)
Requirement already satisfied: click in /usr/local/lib/python3.9/dist-packages (from tensorflow-datasets->tf-models-official>=2.5.1->object-detection==0.1) (8.1.3)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.9/dist-packages (from astunparse>=1.6.0->tensorflow~=2.11.0->tf-models-official>=2.5.1->object-detection==0.1) (0
Requirement already satisfied: googleapis-common-protos<2.0dev,>=1.56.2 in /usr/local/lib/python3.9/dist-packages (from google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5->goo
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from google-auth<3.0.0dev,>=1.19.0->google-api-python-client>=1.6.7->tf-models-official>=2.5
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from scikit-learn>=0.21.3->seqeval->tf-models-official>=2.5.1->object-detection==0.1) (3.1.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-packages (from scikit-learn>=0.21.3->seqeval->tf-models-official>=2.5.1->object-detection==0.1) (1.1.1)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from tensorboard<2.12,>=2.11->tensorflow~=2.11.0->tf-models-official>=2.5.1->object-detection==0.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.9/dist-packages (from tensorboard<2.12,>=2.11->tensorflow~=2.11.0->tf-models-official>=2.5.1->
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.9/dist-packages (from tensorboard<2.12,>=2.11->tensorflow~=2.11.0->tf-models-official>=2.5.1->object-de
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.9/dist-packages (from tensorboard<2.12,>=2.11->tensorflow~=2.11.0->tf-models-official>=2.5.1->object-detection==0.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.9/dist-packages (from tensorboard<2.12,>=2.11->tensorflow~=2.11.0->tf-models-official>=2.5.1->object
Requirement already satisfied: importlib-metadata>=3.6 in /usr/local/lib/python3.9/dist-packages (from typeguard>=2.7->tensorflow-addons->tf-models-official>=2.5.1->object-detection==0.1) (
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.9/dist-packages (from python-slugify->kaggle>=1.3.9->tf-models-official>=2.5.1->object-detection==0.1) (1.3)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.9/dist-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.12,>=2.11->tensorflow~=2.11.0->tf-mode
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.9/dist-packages (from werkzeug>=1.0.1->tensorboard<2.12,>=2.11->tensorflow~=2.11.0->tf-models-official>=2.5.1->obje
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.9/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.12,>=2.11->tensorflo
Building wheels for collected packages: object-detection, avro-python3, crcmod, dill, seqeval, docopt
Building wheel for object-detection (setup.py) ... done
Created wheel for object-detection: filename=object_detection-0.1-py3-none-any.whl size=1697012 sha256=161a1b53b49cd79468ce916ac2c6c5fbc4015e4b4728d43e3349ad38ec113976
Stored in directory: /tmp/pip-ephem-wheel-cache-9b4gm24w/wheels/13/dc/b4/ca25040453ba296d853cb0ccf7c827a599f84e993647b41f42
Building wheel for avro-python3 (setup.py) ... done
Created wheel for avro-python3: filename=avro_python3-1.10.2-py3-none-any.whl size=44008 sha256=8494ff2a27568294f1e6cc37f6073ff30c13d0165b4da3d5b8d3a1d36e786f59
Stored in directory: /root/.cache/pip/wheels/5a/29/4d/510c0e098c49c5e49519f430481a5425e60b8752682d7b1e55
Building wheel for crcmod (setup.py) ... done
Created wheel for crcmod: filename=crcmod-1.7-cp39-cp39-linux_x86_64.whl size=36918 sha256=1afc24df13549c108afc0d237d9ed10453656abf5748d4a2918f429a106eceef
Stored in directory: /root/.cache/pip/wheels/4a/6c/a6/ffdd136310039bf226f2707a9a8e6857be7d70a3fc061f6b36

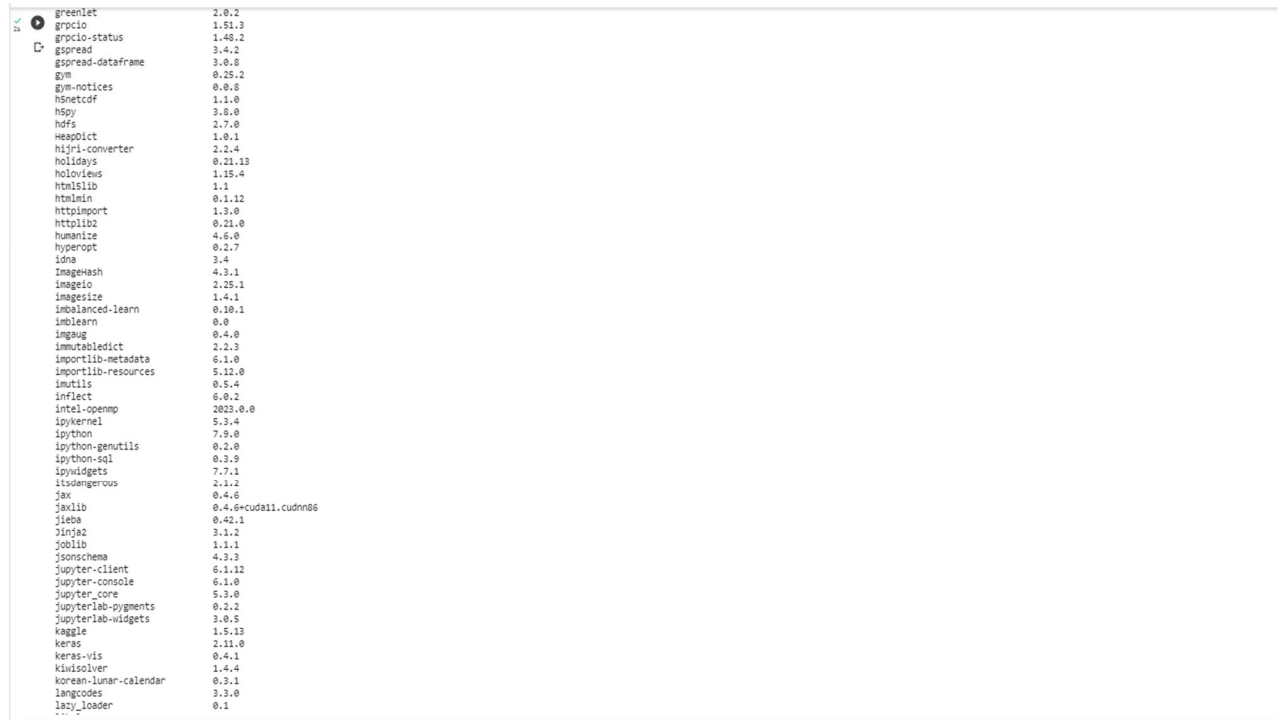


```
Package                       Version
----------------------------- --------------------
absl-py                       1.4.0
alabaster                     0.7.13
albumentations                1.2.1
altair                        4.2.2
apache-beam                   2.46.0
appdirs                       1.4.4
argon2-cffi                   21.3.0
argon2-cffi-bindings          21.2.0
arviz                         0.15.1
astropy                       5.2.1
astunparse                    1.6.3
atomicwrites                  1.4.1
attrs                         22.2.0
audioread                     3.0.0
autograd                      1.5
avro-python3                  1.10.2
Babel                         2.12.1
backcall                      0.2.0
beautifulsoup4                4.11.2
bleach                        6.0.0
blis                          0.7.9
bokeh                         2.4.3
branca                        0.6.0
bs4                           0.0.1
CacheControl                  0.12.11
cachetools                    5.3.0
catalogue                     2.0.8
certifi                       2022.12.7
cffi                          1.15.1
chardet                       3.0.4
charset-normalizer            2.0.12
```

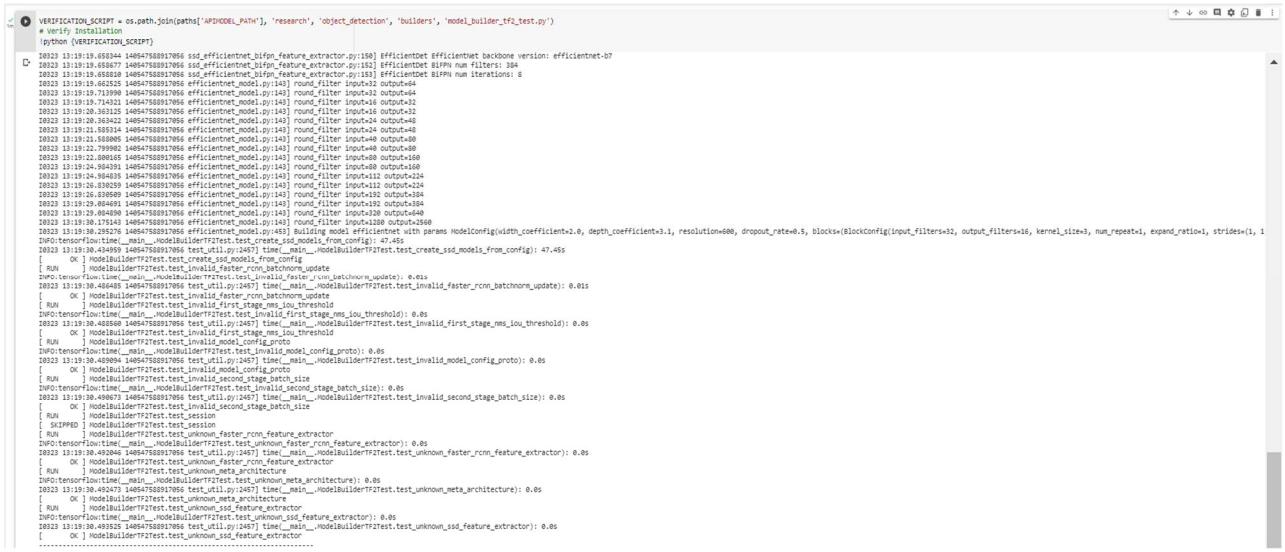


+ Code + Text

dopamine-rl	1.0.5
earthengine-api	0.1.345
easydict	1.10
ecos	2.0.12
editdistance	0.5.3
en-core-web-sm	3.5.0
entrypoints	0.4
ephem	4.1.4
et-xmlfile	1.1.0
etils	1.1.1
etuples	0.3.8
fastai	2.7.11
fastavro	1.7.3
fastcore	1.5.28
fastdownload	0.0.7
fasteners	0.18
fastjsonschema	2.16.3
fastprogress	1.0.3
fastrlock	0.8.1
filelock	3.10.0
firebase-admin	5.3.0
fix-yahoo-finance	0.0.22
Flask	2.2.3
flatbuffers	23.3.3
folium	0.14.0
fonttools	4.39.2
fsspec	2023.3.0
future	0.18.3
gast	0.4.0
GDAL	3.3.2
gdown	4.6.4
gensim	4.3.1
geographiclib	2.0
geopy	2.3.0
gin-config	0.5.0
glob2	0.7
google	2.0.3
google-api-core	2.11.0
google-api-python-client	2.70.0
google-auth	2.16.2
google-auth-httplib2	0.1.0
google-auth-oauthlib	0.4.6
google-cloud-bigquery	3.4.2
google-cloud-bigquery-storage	2.19.0
google-cloud-core	2.3.2
google-cloud-datastore	2.11.1
google-cloud-firestore	2.7.3
google-cloud-language	2.6.1
google-cloud-storage	2.7.0
google-cloud-translate	3.8.4
google-colab	1.0.0
google-crc32c	1.5.0
google-pasta	0.2.0
google-resumable-media	2.4.1
googleapis-common-protos	1.58.0
googledrivedownloader	0.4

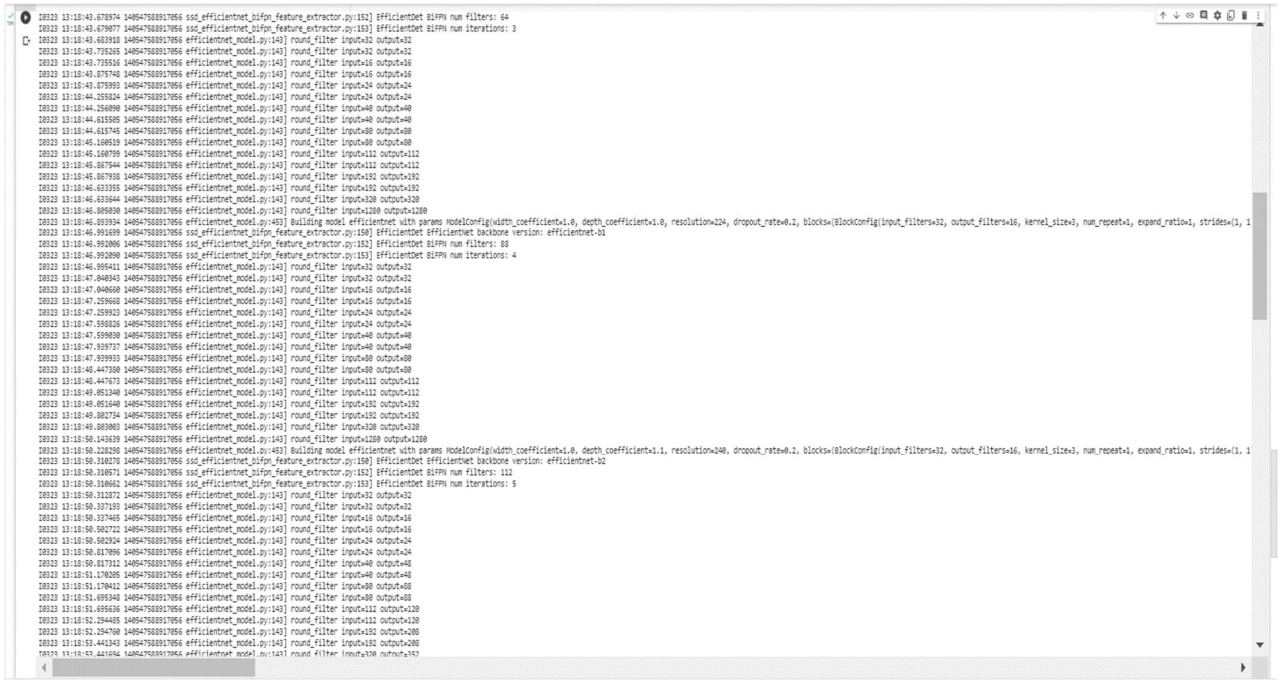


greenlet	2.0.2
grpcio	1.51.3
grpcio-status	1.48.2
gspread	3.4.2
gspread-dataframe	3.0.8
gym	0.25.2
gym-notices	0.0.8
h5netcdf	1.1.0
h5py	3.8.0
hdfs	2.7.0
HeapDict	1.0.1
hijri-converter	2.2.4
holidays	0.21.13
holoviews	1.15.4
html5lib	1.1
htmlmin	0.1.12
httpimport	1.3.0
httplib2	0.21.0
humanize	4.6.0
hyperopt	0.2.7
idna	3.4
ImageHash	4.3.1
imageio	2.25.1
imagesize	1.4.1
imbalanced-learn	0.10.1
imblearn	0.0
imgaug	0.4.0
immutabledict	2.2.3
importlib-metadata	6.1.0
importlib-resources	5.12.0
imutils	0.5.4
inflect	6.0.2
intel-openmp	2023.0.0
ipykernel	5.3.4
ipython	7.9.0
ipython-genutils	0.2.0
ipython-sql	0.3.9
ipywidgets	7.7.1
itsdangerous	2.1.2
jax	0.4.6
jaxlib	0.4.6+cuda11.cudnn86
jieba	0.42.1
Jinja2	3.1.2
joblib	1.1.1
jsonschema	4.3.3
jupyter-client	6.1.12
jupyter-console	6.1.0
jupyter_core	5.3.0
jupyterlab-pygments	0.2.2
jupyterlab-widgets	3.0.5
kaggle	1.5.13
keras	2.11.0
keras-vis	0.4.1
kiwisolver	1.4.4
korean-lunar-calendar	0.3.1
langcodes	3.3.0
lazy_loader	0.1



```
VERIFICATION_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection', 'builders', 'model_builder_tf2_test.py')
# Verify Installation
!python {VERIFICATION_SCRIPT}
```

```
I0323 13:19:19.658344 140547588917056 ssd_efficientnet_bifpn_feature_extractor.py:150] EfficientDet efficientnet backbone version: efficientnet-b7
I0323 13:19:19.658677 140547588917056 ssd_efficientnet_bifpn_feature_extractor.py:152] EfficientDet BiFPN num filters: 384
I0323 13:19:19.658810 140547588917056 ssd_efficientnet_bifpn_feature_extractor.py:153] EfficientDet BiFPN num iterations: 8
I0323 13:19:19.662526 140547588917056 efficientnet_model.py:143] round_filter input=32 output=64
I0323 13:19:19.713990 140547588917056 efficientnet_model.py:143] round_filter input=32 output=64
I0323 13:19:19.714321 140547588917056 efficientnet_model.py:143] round_filter input=16 output=32
I0323 13:19:20.363125 140547588917056 efficientnet_model.py:143] round_filter input=16 output=32
I0323 13:19:20.363422 140547588917056 efficientnet_model.py:143] round_filter input=24 output=48
I0323 13:19:21.585314 140547588917056 efficientnet_model.py:143] round_filter input=24 output=48
I0323 13:19:21.588005 140547588917056 efficientnet_model.py:143] round_filter input=40 output=80
I0323 13:19:22.799902 140547588917056 efficientnet_model.py:143] round_filter input=40 output=80
I0323 13:19:22.800165 140547588917056 efficientnet_model.py:143] round_filter input=80 output=160
I0323 13:19:24.984191 140547588917056 efficientnet_model.py:143] round_filter input=80 output=160
I0323 13:19:24.984635 140547588917056 efficientnet_model.py:143] round_filter input=112 output=224
I0323 13:19:26.830159 140547588917056 efficientnet_model.py:143] round_filter input=112 output=224
I0323 13:19:26.830509 140547588917056 efficientnet_model.py:143] round_filter input=192 output=384
I0323 13:19:29.084691 140547588917056 efficientnet_model.py:143] round_filter input=192 output=384
I0323 13:19:29.084890 140547588917056 efficientnet_model.py:143] round_filter input=320 output=640
I0323 13:19:30.175143 140547588917056 efficientnet_model.py:143] round_filter input=1280 output=2560
I0323 13:19:30.295276 140547588917056 efficientnet_model.py:453] Building model efficientnet with params ModelConfig(width_coefficient=2.0, depth_coefficient=3.1, resolution=600, dropout_rate=0.5, blocks=(BlockConfig(input_filters=32, output_filters=16, kernel_size=3, num_repeat=1, expand_ratio=1, strides=(1, 1
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_create_ssd_models_from_config): 47.45s
I0323 13:19:30.434959 140547588917056 test_util.py:2457] time(__main__.ModelBuilderTF2Test.test_create_ssd_models_from_config): 47.46s
[       OK ] ModelBuilderTF2Test.test_create_ssd_models_from_config
[ RUN      ] ModelBuilderTF2Test.test_invalid_faster_rcnn_batchnorm_update
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_invalid_faster_rcnn_batchnorm_update): 0.01s
I0323 13:19:30.486485 140547588917056 test_util.py:2457] time(__main__.ModelBuilderTF2Test.test_invalid_faster_rcnn_batchnorm_update): 0.01s
[       OK ] ModelBuilderTF2Test.test_invalid_faster_rcnn_batchnorm_update
[ RUN      ] ModelBuilderTF2Test.test_invalid_first_stage_nms_iou_threshold
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_invalid_first_stage_nms_iou_threshold): 0.0s
I0323 13:19:30.488560 140547588917056 test_util.py:2457] time(__main__.ModelBuilderTF2Test.test_invalid_first_stage_nms_iou_threshold): 0.0s
[       OK ] ModelBuilderTF2Test.test_invalid_first_stage_nms_iou_threshold
[ RUN      ] ModelBuilderTF2Test.test_invalid_model_config_proto
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_invalid_model_config_proto): 0.0s
I0323 13:19:30.489094 140547588917056 test_util.py:2457] time(__main__.ModelBuilderTF2Test.test_invalid_model_config_proto): 0.0s
[       OK ] ModelBuilderTF2Test.test_invalid_model_config_proto
[ RUN      ] ModelBuilderTF2Test.test_invalid_second_stage_batch_size
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_invalid_second_stage_batch_size): 0.0s
I0323 13:19:30.490673 140547588917056 test_util.py:2457] time(__main__.ModelBuilderTF2Test.test_invalid_second_stage_batch_size): 0.0s
[       OK ] ModelBuilderTF2Test.test_invalid_second_stage_batch_size
[ RUN      ] ModelBuilderTF2Test.test_session
[  SKIPPED ] ModelBuilderTF2Test.test_session
[ RUN      ] ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor): 0.0s
I0323 13:19:30.492046 140547588917056 test_util.py:2457] time(__main__.ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor): 0.0s
[       OK ] ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor
[ RUN      ] ModelBuilderTF2Test.test_unknown_meta_architecture
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_unknown_meta_architecture): 0.0s
I0323 13:19:30.492473 140547588917056 test_util.py:2457] time(__main__.ModelBuilderTF2Test.test_unknown_meta_architecture): 0.0s
[       OK ] ModelBuilderTF2Test.test_unknown_meta_architecture
[ RUN      ] ModelBuilderTF2Test.test_unknown_ssd_feature_extractor
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_unknown_ssd_feature_extractor): 0.0s
I0323 13:19:30.493525 140547588917056 test_util.py:2457] time(__main__.ModelBuilderTF2Test.test_unknown_ssd_feature_extractor): 0.0s
[       OK ] ModelBuilderTF2Test.test_unknown_ssd_feature_extractor
----------------------------------------------------------------------
```



```
I0323 13:18:43.678974 140547588917056 ssd_efficientnet_bifpn_feature_extractor.py:152] EfficientDet BiFPN num filters: 64
I0323 13:18:43.679077 140547588917056 ssd_efficientnet_bifpn_feature_extractor.py:153] EfficientDet BiFPN num iterations: 3
I0323 13:18:43.683918 140547588917056 efficientnet_model.py:143] round_filter input=32 output=32
I0323 13:18:43.735265 140547588917056 efficientnet_model.py:143] round_filter input=32 output=32
I0323 13:18:43.735516 140547588917056 efficientnet_model.py:143] round_filter input=16 output=16
I0323 13:18:43.875748 140547588917056 efficientnet_model.py:143] round_filter input=16 output=16
I0323 13:18:43.875993 140547588917056 efficientnet_model.py:143] round_filter input=24 output=24
I0323 13:18:44.255824 140547588917056 efficientnet_model.py:143] round_filter input=24 output=24
I0323 13:18:44.256090 140547588917056 efficientnet_model.py:143] round_filter input=40 output=40
I0323 13:18:44.615505 140547588917056 efficientnet_model.py:143] round_filter input=40 output=40
I0323 13:18:44.615745 140547588917056 efficientnet_model.py:143] round_filter input=80 output=80
I0323 13:18:45.160519 140547588917056 efficientnet_model.py:143] round_filter input=80 output=80
I0323 13:18:45.160799 140547588917056 efficientnet_model.py:143] round_filter input=112 output=112
I0323 13:18:45.867544 140547588917056 efficientnet_model.py:143] round_filter input=112 output=112
I0323 13:18:45.867938 140547588917056 efficientnet_model.py:143] round_filter input=192 output=192
I0323 13:18:46.633355 140547588917056 efficientnet_model.py:143] round_filter input=192 output=192
I0323 13:18:46.633644 140547588917056 efficientnet_model.py:143] round_filter input=320 output=320
I0323 13:18:46.805830 140547588917056 efficientnet_model.py:143] round_filter input=1280 output=1280
I0323 13:18:46.893934 140547588917056 efficientnet_model.py:453] Building model efficientnet with params ModelConfig(width_coefficient=1.0, depth_coefficient=1.0, resolution=224, dropout_rate=0.2, blocks=(BlockConfig(input_filters=32, output_filters=16, kernel_size=3, num_repeat=1, expand_ratio=1, strides=(1, 1
I0323 13:18:46.991699 140547588917056 ssd_efficientnet_bifpn_feature_extractor.py:150] EfficientDet efficientnet backbone version: efficientnet-b1
I0323 13:18:46.992006 140547588917056 ssd_efficientnet_bifpn_feature_extractor.py:152] EfficientDet BiFPN num filters: 88
I0323 13:18:46.992090 140547588917056 ssd_efficientnet_bifpn_feature_extractor.py:153] EfficientDet BiFPN num iterations: 4
I0323 13:18:46.995411 140547588917056 efficientnet_model.py:143] round_filter input=32 output=32
I0323 13:18:47.040343 140547588917056 efficientnet_model.py:143] round_filter input=32 output=32
I0323 13:18:47.040608 140547588917056 efficientnet_model.py:143] round_filter input=16 output=16
I0323 13:18:47.259668 140547588917056 efficientnet_model.py:143] round_filter input=16 output=16
I0323 13:18:47.259923 140547588917056 efficientnet_model.py:143] round_filter input=24 output=24
I0323 13:18:47.586826 140547588917056 efficientnet_model.py:143] round_filter input=24 output=24
I0323 13:18:47.599030 140547588917056 efficientnet_model.py:143] round_filter input=40 output=40
I0323 13:18:47.919717 140547588917056 efficientnet_model.py:143] round_filter input=40 output=40
I0323 13:18:47.919933 140547588917056 efficientnet_model.py:143] round_filter input=80 output=80
I0323 13:18:48.447300 140547588917056 efficientnet_model.py:143] round_filter input=80 output=80
I0323 13:18:48.447673 140547588917056 efficientnet_model.py:143] round_filter input=112 output=112
I0323 13:18:49.051340 140547588917056 efficientnet_model.py:143] round_filter input=112 output=112
I0323 13:18:49.051640 140547588917056 efficientnet_model.py:143] round_filter input=192 output=192
I0323 13:18:49.802734 140547588917056 efficientnet_model.py:143] round_filter input=192 output=192
I0323 13:18:49.803003 140547588917056 efficientnet_model.py:143] round_filter input=320 output=320
I0323 13:18:50.143639 140547588917056 efficientnet_model.py:143] round_filter input=1280 output=1280
I0323 13:18:50.228298 140547588917056 efficientnet_model.py:453] Building model efficientnet with params ModelConfig(width_coefficient=1.0, depth_coefficient=1.1, resolution=240, dropout_rate=0.2, blocks=(BlockConfig(input_filters=32, output_filters=16, kernel_size=3, num_repeat=1, expand_ratio=1, strides=(1, 1
I0323 13:18:50.310278 140547588917056 ssd_efficientnet_bifpn_feature_extractor.py:150] EfficientDet efficientnet backbone version: efficientnet-b2
I0323 13:18:50.310571 140547588917056 ssd_efficientnet_bifpn_feature_extractor.py:152] EfficientDet BiFPN num filters: 112
I0323 13:18:50.310662 140547588917056 ssd_efficientnet_bifpn_feature_extractor.py:153] EfficientDet BiFPN num iterations: 5
I0323 13:18:50.312872 140547588917056 efficientnet_model.py:143] round_filter input=32 output=32
I0323 13:18:50.337193 140547588917056 efficientnet_model.py:143] round_filter input=32 output=32
I0323 13:18:50.337465 140547588917056 efficientnet_model.py:143] round_filter input=16 output=16
I0323 13:18:50.502722 140547588917056 efficientnet_model.py:143] round_filter input=16 output=16
I0323 13:18:50.502924 140547588917056 efficientnet_model.py:143] round_filter input=24 output=24
I0323 13:18:50.817096 140547588917056 efficientnet_model.py:143] round_filter input=24 output=24
I0323 13:18:50.817312 140547588917056 efficientnet_model.py:143] round_filter input=40 output=40
I0323 13:18:51.170205 140547588917056 efficientnet_model.py:143] round_filter input=40 output=40
I0323 13:18:51.170412 140547588917056 efficientnet_model.py:143] round_filter input=80 output=80
I0323 13:18:51.695348 140547588917056 efficientnet_model.py:143] round_filter input=80 output=80
I0323 13:18:51.695636 140547588917056 efficientnet_model.py:143] round_filter input=112 output=120
I0323 13:18:52.294485 140547588917056 efficientnet_model.py:143] round_filter input=112 output=120
I0323 13:18:52.294760 140547588917056 efficientnet_model.py:143] round_filter input=192 output=200
I0323 13:18:53.441343 140547588917056 efficientnet_model.py:143] round_filter input=192 output=200
```

```
I0323 13:19:19.658677 140547588917056 ssd_efficientnet_bifpn_feature_extractor.py:152] EfficientDet BiFPN num filters: 384
I0323 13:19:19.658810 140547588917056 ssd_efficientnet_bifpn_feature_extractor.py:153] EfficientDet BiFPN num iterations: 8
I0323 13:19:19.662525 140547588917056 efficientnet_model.py:143] round_filter input=32 output=64
I0323 13:19:19.713990 140547588917056 efficientnet_model.py:143] round_filter input=32 output=64
I0323 13:19:19.714321 140547588917056 efficientnet_model.py:143] round_filter input=16 output=32
I0323 13:19:20.363125 140547588917056 efficientnet_model.py:143] round_filter input=16 output=32
I0323 13:19:20.363422 140547588917056 efficientnet_model.py:143] round_filter input=24 output=48
I0323 13:19:21.585314 140547588917056 efficientnet_model.py:143] round_filter input=24 output=48
I0323 13:19:21.588005 140547588917056 efficientnet_model.py:143] round_filter input=40 output=80
I0323 13:19:22.799902 140547588917056 efficientnet_model.py:143] round_filter input=40 output=80
I0323 13:19:22.800165 140547588917056 efficientnet_model.py:143] round_filter input=80 output=160
I0323 13:19:24.984391 140547588917056 efficientnet_model.py:143] round_filter input=80 output=160
I0323 13:19:24.984835 140547588917056 efficientnet_model.py:143] round_filter input=112 output=224
I0323 13:19:26.830259 140547588917056 efficientnet_model.py:143] round_filter input=112 output=224
I0323 13:19:26.830509 140547588917056 efficientnet_model.py:143] round_filter input=192 output=384
I0323 13:19:29.084691 140547588917056 efficientnet_model.py:143] round_filter input=192 output=384
I0323 13:19:29.084890 140547588917056 efficientnet_model.py:143] round_filter input=320 output=640
I0323 13:19:30.175143 140547588917056 efficientnet_model.py:143] round_filter input=1280 output=2560
I0323 13:19:30.295276 140547588917056 efficientnet_model.py:453] Building model efficientnet with params ModelConfig(width_coefficient=2.0, depth_coefficient=3.1, resolution=600, dropout_rate=0.5, blocks=(BlockConfig(input_filters=32, output_filters=16, kernel
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_create_ssd_models_from_config): 47.45s
I0323 13:19:30.434859 140547588917056 test_util.py:2457] time(__main__.ModelBuilderTF2Test.test_create_ssd_models_from_config): 47.45s
[       OK ] ModelBuilderTF2Test.test_create_ssd_models_from_config
[ RUN      ] ModelBuilderTF2Test.test_invalid_faster_rcnn_batchnorm_update
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_invalid_faster_rcnn_batchnorm_update): 0.01s
I0323 13:19:30.486485 140547588917056 test_util.py:2457] time(__main__.ModelBuilderTF2Test.test_invalid_faster_rcnn_batchnorm_update): 0.01s
[       OK ] ModelBuilderTF2Test.test_invalid_faster_rcnn_batchnorm_update
[ RUN      ] ModelBuilderTF2Test.test_invalid_first_stage_nms_iou_threshold
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_invalid_first_stage_nms_iou_threshold): 0.0s
I0323 13:19:30.488560 140547588917056 test_util.py:2457] time(__main__.ModelBuilderTF2Test.test_invalid_first_stage_nms_iou_threshold): 0.0s
[       OK ] ModelBuilderTF2Test.test_invalid_first_stage_nms_iou_threshold
[ RUN      ] ModelBuilderTF2Test.test_invalid_model_config_proto
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_invalid_model_config_proto): 0.0s
I0323 13:19:30.489094 140547588917056 test_util.py:2457] time(__main__.ModelBuilderTF2Test.test_invalid_model_config_proto): 0.0s
[       OK ] ModelBuilderTF2Test.test_invalid_model_config_proto
[ RUN      ] ModelBuilderTF2Test.test_invalid_second_stage_batch_size
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_invalid_second_stage_batch_size): 0.0s
I0323 13:19:30.490673 140547588917056 test_util.py:2457] time(__main__.ModelBuilderTF2Test.test_invalid_second_stage_batch_size): 0.0s
[       OK ] ModelBuilderTF2Test.test_invalid_second_stage_batch_size
[ RUN      ] ModelBuilderTF2Test.test_session
[  SKIPPED ] ModelBuilderTF2Test.test_session
[ RUN      ] ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor): 0.0s
I0323 13:19:30.492046 140547588917056 test_util.py:2457] time(__main__.ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor): 0.0s
[       OK ] ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor
[ RUN      ] ModelBuilderTF2Test.test_unknown_meta_architecture
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_unknown_meta_architecture): 0.0s
I0323 13:19:30.492473 140547588917056 test_util.py:2457] time(__main__.ModelBuilderTF2Test.test_unknown_meta_architecture): 0.0s
[       OK ] ModelBuilderTF2Test.test_unknown_meta_architecture
[ RUN      ] ModelBuilderTF2Test.test_unknown_ssd_feature_extractor
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_unknown_ssd_feature_extractor): 0.0s
I0323 13:19:30.493525 140547588917056 test_util.py:2457] time(__main__.ModelBuilderTF2Test.test_unknown_ssd_feature_extractor): 0.0s
[       OK ] ModelBuilderTF2Test.test_unknown_ssd_feature_extractor
----------------------------------------------------------------------
Ran 24 tests in 58.040s

OK (skipped=1)
```

```
[14] import object_detection
```

```
if os.name =='posix':
    !wget {PRETRAINED_MODEL_URL}
    !mv {PRETRAINED_MODEL_NAME+'.tar.gz'} {paths['PRETRAINED_MODEL_PATH']}
    !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf {PRETRAINED_MODEL_NAME+'.tar.gz'}
if os.name == 'nt':
    wget.download(PRETRAINED_MODEL_URL)
    !move {PRETRAINED_MODEL_NAME+'.tar.gz'} {paths['PRETRAINED_MODEL_PATH']}
    !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf {PRETRAINED_MODEL_NAME+'.tar.gz'}
```

```
--2023-03-23 13:19:32--  http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz
Resolving download.tensorflow.org (download.tensorflow.org)... 142.251.16.128, 2607:f8b0:4004:c09::80
Connecting to download.tensorflow.org (download.tensorflow.org)|142.251.16.128|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 20515344 (20M) [application/x-tar]
Saving to: ‘ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz’

ssd_mobilenet_v2_fp 100%[===================>]  19.56M   108MB/s    in 0.2s

2023-03-23 13:19:32 (108 MB/s) - ‘ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz’ saved [20515344/20515344]

ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/
ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint/
ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint/ckpt-0.data-00000-of-00001
ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint/checkpoint
ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint/ckpt-0.index
ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/pipeline.config
ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/saved_model/
ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/saved_model/saved_model.pb
ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/saved_model/variables/
ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/saved_model/variables/variables.data-00000-of-00001
ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/saved_model/variables/variables.index
```

CAPTURING IMAGE IN REAL TIME

number_plate.py

C: > Users > tvasanthi > Downloads > number_plate.py

```python
import cv2

harcascade = "model/haarcascade_russian_plate_number.xml"

cap = cv2.VideoCapture(0)

cap.set(3, 640) # width
cap.set(4, 480) #height

min_area = 500
count = 0

while True:
    success, img = cap.read()

    plate_cascade = cv2.CascadeClassifier(harcascade)
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    plates = plate_cascade.detectMultiScale(img_gray, 1.1, 4)

    for (x,y,w,h) in plates:
        area = w * h

        if area > min_area:
            cv2.rectangle(img, (x,y), (x+w, y+h), (0,255,0), 2)
            cv2.putText(img, "Number Plate", (x,y-5), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (255, 0, 255), 2)

            img_roi = img[y: y+h, x:x+w]
            cv2.imshow("ROI", img_roi)
```

```python
    cv2.imshow("Result", img)

    if cv2.waitKey(1) & 0xFF == ord('s'):
        cv2.imwrite("plates/scaned_img_" + str(count) + ".jpg", img_roi)
        cv2.rectangle(img, (0,200), (640,300), (0,255,0), cv2.FILLED)
        cv2.putText(img, "Plate Saved", (150, 265), cv2.FONT_HERSHEY_COMPLEX_SMALL, 2, (0, 0, 255), 2)
        cv2.imshow("Results",img)
        cv2.waitKey(500)
        count += 1
```


Ford
MH12DE1433
Planet Ford


MH12DE1433

APPLYING OCR TO EXTRACT TEXT

```
[2] ## OCR to TEXT Analysis on the Captured Images
    !pip install opencv-python
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: opencv-python in /usr/local/lib/python3.9/dist-packages (4.7.0.72)
Requirement already satisfied: numpy>=1.19.3 in /usr/local/lib/python3.9/dist-packages (from opencv-python) (1.22.4)
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files 4 files

- **WhatsApp Image 2023-03-23 at 8.33.52 PM.jpeg**(image/jpeg) - 175694 bytes, last modified: 3/23/2023 - 100% done
- **WhatsApp Image 2023-03-23 at 8.33.51 PM (2).jpeg**(image/jpeg) - 111209 bytes, last modified: 3/23/2023 - 100% done
- **WhatsApp Image 2023-03-23 at 8.33.51 PM (1).jpeg**(image/jpeg) - 16334 bytes, last modified: 3/23/2023 - 100% done
- **WhatsApp Image 2023-03-23 at 8.33.51 PM.jpeg**(image/jpeg) - 40046 bytes, last modified: 3/23/2023 - 100% done

```
Saving WhatsApp Image 2023-03-23 at 8.33.52 PM.jpeg to WhatsApp Image 2023-03-23 at 8.33.52 PM.jpeg
Saving WhatsApp Image 2023-03-23 at 8.33.51 PM (2).jpeg to WhatsApp Image 2023-03-23 at 8.33.51 PM (2).jpeg
Saving WhatsApp Image 2023-03-23 at 8.33.51 PM (1).jpeg to WhatsApp Image 2023-03-23 at 8.33.51 PM (1).jpeg
Saving WhatsApp Image 2023-03-23 at 8.33.51 PM.jpeg to WhatsApp Image 2023-03-23 at 8.33.51 PM.jpeg
```

```
!pip install easyocr
!pip install imutils
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting easyocr
  Downloading easyocr-1.6.2-py3-none-any.whl (2.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.9/2.9 MB 18.6 MB/s eta 0:00:00
Collecting opencv-python-headless<=4.5.4.60
  Downloading opencv_python_headless-4.5.4.60-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (47.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 47.6/47.6 MB 10.8 MB/s eta 0:00:00
Collecting ninja
  Downloading ninja-1.11.1-py2.py3-none-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (145 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 146.0/146.0 KB 17.1 MB/s eta 0:00:00
Requirement already satisfied: PyYAML in /usr/local/lib/python3.9/dist-packages (from easyocr) (6.0)
Requirement already satisfied: Pillow in /usr/local/lib/python3.9/dist-packages (from easyocr) (8.4.0)
Requirement already satisfied: torch in /usr/local/lib/python3.9/dist-packages (from easyocr) (1.13.1+cu116)
Collecting python-bidi
  Downloading python_bidi-0.4.2-py2.py3-none-any.whl (30 kB)
Collecting pyclipper
```

```
Requirement already satisfied: torchvision>=0.5 in /usr/local/lib/python3.9/dist-packages (from easyocr) (0.14.1+cu116)
Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-packages (from easyocr) (1.10.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages (from easyocr) (1.22.4)
Requirement already satisfied: scikit-image in /usr/local/lib/python3.9/dist-packages (from easyocr) (0.19.3)
Requirement already satisfied: Shapely in /usr/local/lib/python3.9/dist-packages (from easyocr) (2.0.1)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages (from torchvision>=0.5->easyocr) (2.27.1)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.9/dist-packages (from torchvision>=0.5->easyocr) (4.5.0)
Requirement already satisfied: six in /usr/local/lib/python3.9/dist-packages (from python-bidi->easyocr) (1.16.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-packages (from scikit-image->easyocr) (23.0)
Requirement already satisfied: networkx>=2.2 in /usr/local/lib/python3.9/dist-packages (from scikit-image->easyocr) (3.0)
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.9/dist-packages (from scikit-image->easyocr) (2023.3.15)
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.9/dist-packages (from scikit-image->easyocr) (1.4.1)
Requirement already satisfied: imageio>=2.4.1 in /usr/local/lib/python3.9/dist-packages (from scikit-image->easyocr) (2.25.1)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests->torchvision>=0.5->easyocr) (2022.12.7)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist-packages (from requests->torchvision>=0.5->easyocr) (1.26.15)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests->torchvision>=0.5->easyocr) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests->torchvision>=0.5->easyocr) (3.4)
Installing collected packages: pyclipper, ninja, python-bidi, opencv-python-headless, easyocr
  Attempting uninstall: opencv-python-headless
    Found existing installation: opencv-python-headless 4.7.0.72
    Uninstalling opencv-python-headless-4.7.0.72:
```

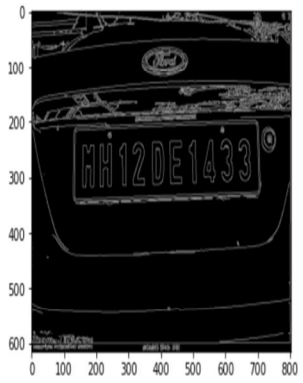
```
import cv2
from matplotlib import pyplot as plt
import numpy as np
import imutils
import easyocr
```

```
[6] img = cv2.imread('d1.JPG')
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    plt.imshow(cv2.cvtColor(gray, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f0ef1ed0940>
```

```
bfilter = cv2.bilateralFilter(gray, 11, 17, 17) #Noise reduction
edged = cv2.Canny(bfilter, 30, 200) #Edge detection
plt.imshow(cv2.cvtColor(edged, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f0ef1de7640>
```




```
[8] keypoints = cv2.findContours(edged.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    contours = imutils.grab_contours(keypoints)
    contours = sorted(contours, key=cv2.contourArea, reverse=True)[:10]
```

```
[9] location = None
    for contour in contours:
        approx = cv2.approxPolyDP(contour, 10, True)
        if len(approx) == 4:
            location = approx
            break
```

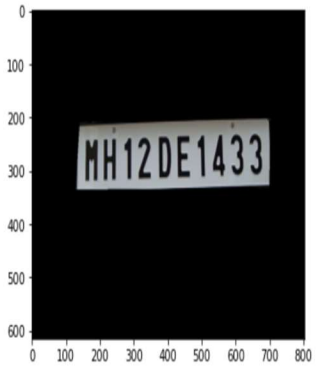
```
[10] location
```

```
array([[[700, 203]],

       [[141, 211]],

       [[129, 335]],

       [[698, 332]]], dtype=int32)
```

```
[11] mask = np.zeros(gray.shape, np.uint8)
     new_image = cv2.drawContours(mask, [location], 0,255, -1)
     new_image = cv2.bitwise_and(img, img, mask=mask)
```

```
[12] plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f0ef1de71c0>
```




```
[13] (x,y) = np.where(mask==255)
     (x1, y1) = (np.min(x), np.min(y))
     (x2, y2) = (np.max(x), np.max(y))
     cropped_image = gray[x1:x2+1, y1:y2+1]
```

```
[14] plt.imshow(cv2.cvtColor(cropped_image, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f0e3e6a6be0>
```


```
[15] reader = easyocr.Reader(['en'])
     result = reader.readtext(cropped_image)
     result
```

```
WARNING:easyocr.easyocr:CUDA not available - defaulting to CPU. Note: This module is much faster with a GPU.
WARNING:easyocr.easyocr:Downloading detection model, please wait. This may take several minutes depending upon your network connection.
Progress: |████████████████████████████████████████| 100.0% CompleteWARNING:easyocr.easyocr:Downloading recognition model, please wait. This may take several minutes dependir
Progress: |████████████████████████████████████████| 100.0% Complete[([[13, 13], [564, 13], [564, 130], [13, 130]],
  'HHI2 DE1433',
  0.4889769776179334)]
```

```
text = result[0][-2]
font = cv2.FONT_HERSHEY_SIMPLEX
res = cv2.putText(img, text=text, org=(approx[0][0][0], approx[1][0][1]+60), fontFace=font, fontScale=1, color=(0,255,0), thickness=2, lineType=cv2.LINE_AA)
res = cv2.rectangle(img, tuple(approx[0][0]), tuple(approx[2][0]), (0,255,0),3)
plt.imshow(cv2.cvtColor(res, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f0e3bcd2280>
```

8. RESULTS AND DISCUSSIONS

Some ANPR systems may use easy image processing methods to identify predictable license plate designs. A few examples of the specialized object detectors used by advanced ANPR systems are HOG, CNN, SVM, and YOLO.

ANPR machines that are more advanced and intelligent today employ ANPR software with AI capabilities. Neural network methods serve as its foundation. In ANPR, like in many other fields, computer vision and machine learning have their benefits.

Because to the enormous variety of license plate types seen throughout different areas, states, and countries, ANPR is challenging. Identification of number plates is made much more difficult by the requirement that any ANPR system function in real time. Because of this, ML, CV, and AI techniques can significantly improve ANPR.

The papers on the list covered various stages of ANPR system techniques. provided new researchers with a strong library of references in the field of license plate detection. Nevertheless, the research did not compare the rates of accuracy of various recognition systems. tested the efficacy of many vintage algorithms.

The results revealed that the ANPR system's efficacy is unstable and that performance varies as a result of a number of ANPR-affecting factors, including noise, ambient circumstances, the choice of the used algorithms, and model training.

9. CONCLUSION & FUTURE SCOPE

Conclusion:

By the use of digital image processing, this research enables the identification of vehicle licence plates. This method makes it simple for one person to efficiently monitor traffic and locate it. I was able to quickly obtain a number of findings from it, including: • If the registered car is banned. movable objects engaged in traffic.
• The system is more effective when data storage and transport are simplified.

The system was created utilizing a modular strategy, making upgrades simple. the ability to change out various sub-modules to adapt them for use with huge viewing surfaces. System performance sets it apart from the competition, particularly when application costs must be maintained to a minimum. Additionally, it is quite adaptable and versatile due to the modular construction. By the use of digital image processing, this research enables the identification of vehicle licence plates. This method makes it simple for one person to efficiently monitor traffic and locate it. I was able to quickly obtain a number of findings from it, including: • If the registered car is banned. movable objects engaged in traffic.
• The system is more effective when data storage and transport are simplified.

The system was created utilizing a modular strategy, making upgrades simple. the ability to change out various sub-modules to adapt them for use with huge viewing surfaces. System performance sets it apart from the competition, particularly when application costs must be maintained to a minimum. Additionally, it is quite adaptable and versatile due to the modular construction.

The identification of plates may take a while with ANPR devices since they depend on sophisticated optical, processing, and digitalization capabilities. The ANPR solutions that are currently available don't offer a standardized set for all the countries; rather, each company must be given a well-optimized system for different parts/regions of the world because the same system that is developed needs to be tailored to the region where it will be used, taking all relevant factors into account. OCR software typically has regional adaptations. It must be confirmed that the library or engine that is installed on the camera supports the necessary nations. Each ANPR solution system offered by manufacturers has unique advantages and disadvantages.

Future's work:

A future application area is automotive vehicle detection systems play an important role in threat detection defense. It can also improve safety in relationships with women. Because you can easily check the license plate before use taxi. The scheme can be made more robust in the presence of radiation.
and a sharp camera is used. The government should do something that are interested in the invention of this system because it does not cost money and is environmentally friendly if it is used effectively in various fields.

Automatic licence plate recognition is the foundation of this project, and it has been noted that current technology pays little attention to increasing the system's efficiency in terms of power consumption.

When implementing this system in a real-time application, the ability to use high-definition cameras to improve overall accuracy can make the system more robust. Additionally, the sensor can be designed so that the camera only captures images when necessary to save energy.

10. REFERENCES

[1] Amninder Kaur, Sonika Jindal, Richa Jindal "License Plate Recognition Using Support Vector Machine (SVM)" Dept. Of Computer Science, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 7.

[2] ANISH LAZRUS, SIDDHARTHA CHOUBEY, SINHA G.R.,"AN EFFICIENT METHOD OF VEHICLE NUMBER PLATE DETECTION AND RECOGNITION" Department of Computer Science, International Journal of Machine Intelligence, Volume 3, Issue 3.

[3] Abhay Singh, Anand Kumar Gupta, Anmol Singh, Anuj Gupta, Sherish Johri, "VEHICLE NUMBER PLATE DETECTION USING IMAGE PROCESSING", Department of IT, Volume: 05 Issue: 03 | Mar-2018.

[4] Ganesh R. Jadhav, Kailash J. Karande, "Automatic Vehicle Number Plate Recognition for Vehicle Parking Management System", IISTE, Vol.5, No.11, 2014.

[5] Mutua Simon Mandi, Bernard Shibwabo, Kaibiru Mutua Raphael," An Automatic Number Plate Recognition System for Car Park Management", International Journal of Computer Applications, Volume 175 – No.7, October 2017.

[6] https://en.wikipedia.org/wiki/Automatic_number-plate_recognition.

[7] Du, S.; Ibrahim, M.; Shehata, M.; Badawy, W. Automatic license plate recognition (ALPR): A state-of-the-art review. IEEE Trans. Circuits Syst. Video Technol. 2012,23, 311–325.

[8] Birgillito, G.; Rindone, C.; Vitetta, A. Passenger mobility in a discontinuous space: Modeling access/egress to maritime barrier in a case study. J. Adv. Transp. 2018,2018, 6518329.