

# **“TO DEVELOP CNN BASED VEHICLE DAMAGE DETECTION MODEL”**

**BY :- Rohan Salunkhe**

The background features a dark charcoal grey color. There are three large, stylized teal triangles: one pointing down in the top right, and two pointing up in the bottom left and bottom right corners. The triangle in the bottom left is split vertically, with a darker teal on the left and a lighter teal on the right. The triangle in the bottom right is also split vertically, with a lighter teal on the left and a darker teal on the right.

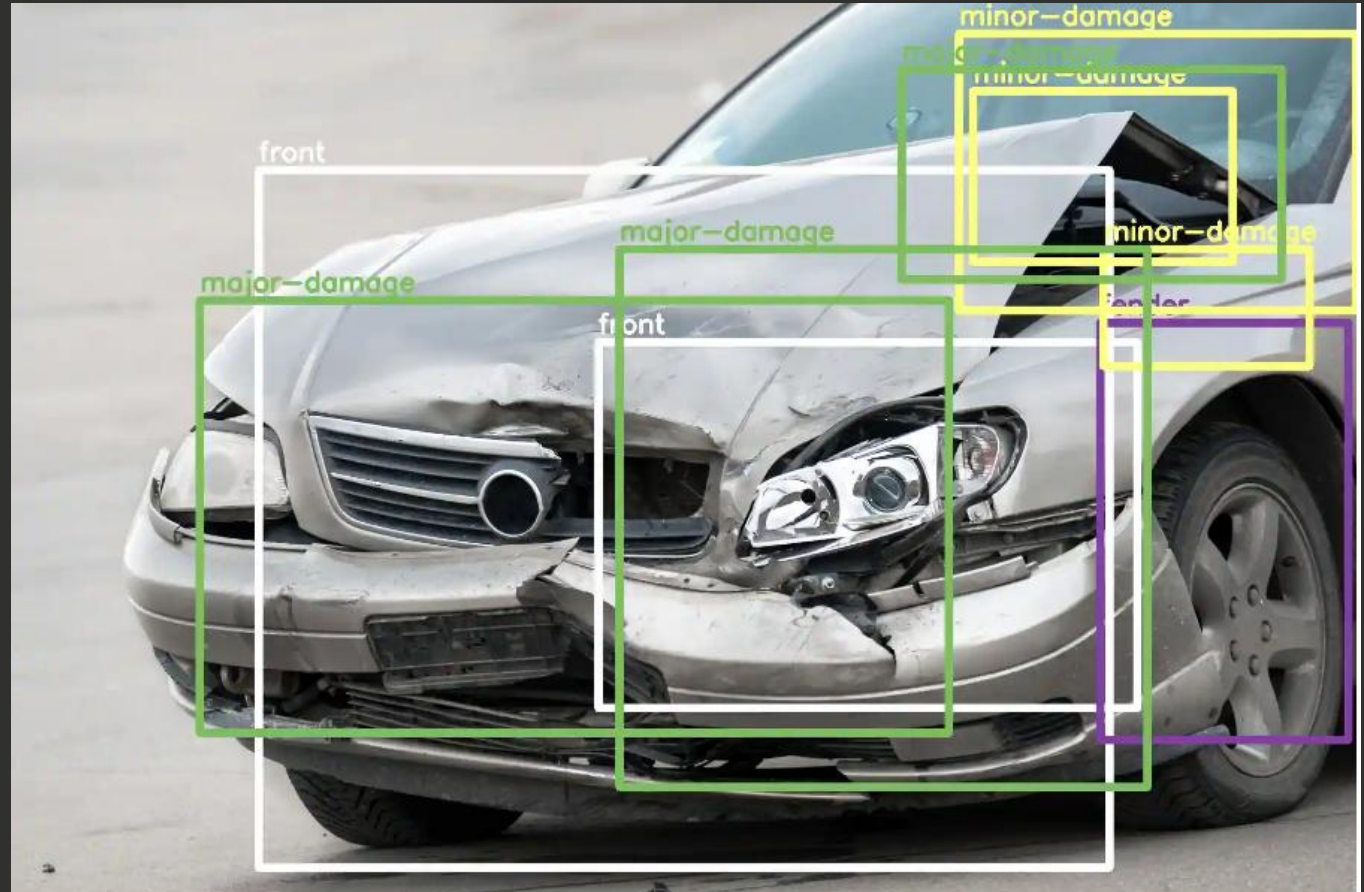
# 1

## PART ONE

Introduction, Problem Statement

# Introduction

Car damage detection is an essential aspect of vehicle maintenance and repair, aimed at identifying and assessing damage to a vehicle's surface. Traditionally, this process has been carried out manually, which involves visual inspection by technicians or the use of specialized tools. However, manual methods are often time-consuming, subjective, and prone to human



# Problem Statement

**To develop CNN based damage detection model :-** The current manual and basic image processing methods for detecting vehicle dents are inefficient and inaccurate, highlighting the need for an automated system using advanced machine learning techniques, such as CNNs, to ensure fast, precise, and reliable damage detection.



The background is a dark charcoal grey. There are three large, stylized triangles in shades of teal. One is in the top right corner, pointing downwards. Another is in the bottom left corner, pointing upwards. A third is partially visible on the right edge, also pointing downwards. The number '2' is a large, white, sans-serif digit.

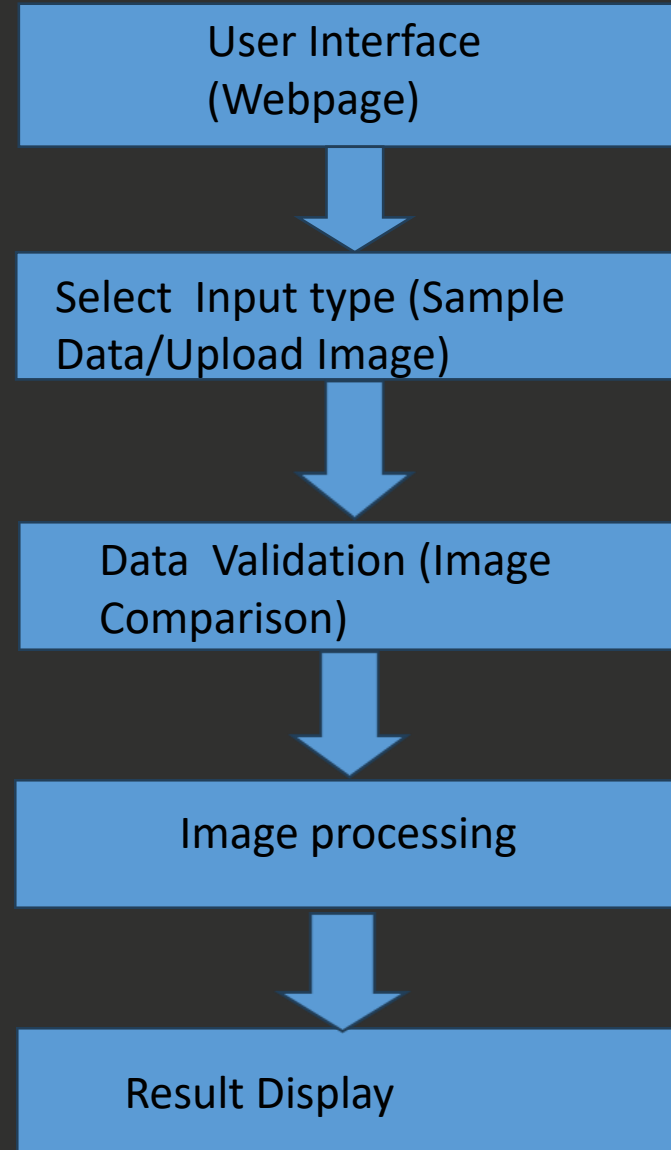
# 2

## PART TWO

Block Diagram, User Interface, User  
benefits, Libraries

## Implementation of The Project

# Block Diagram of Vehical Detection





# User Interface

**Main Dashboard** - Overview of the model status (e.g., "Ready," "Processing," "Complete").  
- Quick access buttons for common tasks (e.g., "Upload Image," "View Results," "Settings").

- **Image Upload Section** :- Drag-and-drop area for image upload.
  - Browse button to select images from the device.
  - Supported formats and size limitations displayed.
- **Processing Area** :- Progress bar indicating the status of the image processing.
  - Notifications for errors or issues during processing.
- **Results Display** :- Image display showing the uploaded car image.
  - An overlay or highlight indicating detected dents.
  - Summary of results (e.g., number of dents detected, severity levels).
- **Detailed Analysis** :- Option to view detailed metrics or statistics about the detection (e.g., size, depth of dents).
  - Option to save or export results (PDF, image).



## User benefits

- **Cost Efficiency:** Early detection of dents can lead to timely repairs, reducing repair costs and preventing further damage.
- **Quality Control:** For manufacturers and dealerships, such a model can ensure that vehicles meet quality standards before delivery.
- **Insurance Claims:** It can streamline the claims process by providing accurate assessments of damage.
- **Enhanced Customer Service:** Quick and accurate dent detection improves customer satisfaction by reducing wait times for assessments and repairs.
- **Data Insights:** Collecting data on dent occurrences can help in understanding patterns, leading to better designs or protective measures.
- **Automation:** Automating the detection process can save time and reduce human error in inspections..



# Libraries And Techniques

- **Keras :**

Keras is an open source Neural network Libraries Written in Python that runs on the top of Tensorflow .It is designed to be modular, Fast and easy to use. Keras High-level API handles the way we make models, defining layers, or set up to multiple input-output model.

- **Tensor Flow:**

It is an open source artificial intelligence library, using data flow graphs to build models .it allows developers to create large scale neural networks with many layers. Tensorflow is mainly used for: Classification, Perception, Understanding, Discovering, Prediction and creation.

- **PyTorch:**

Known for its dynamic computational graph and ease of use, PyTorch is a popular choice for research and prototyping. It provides a flexible interface for defining and training neural networks.

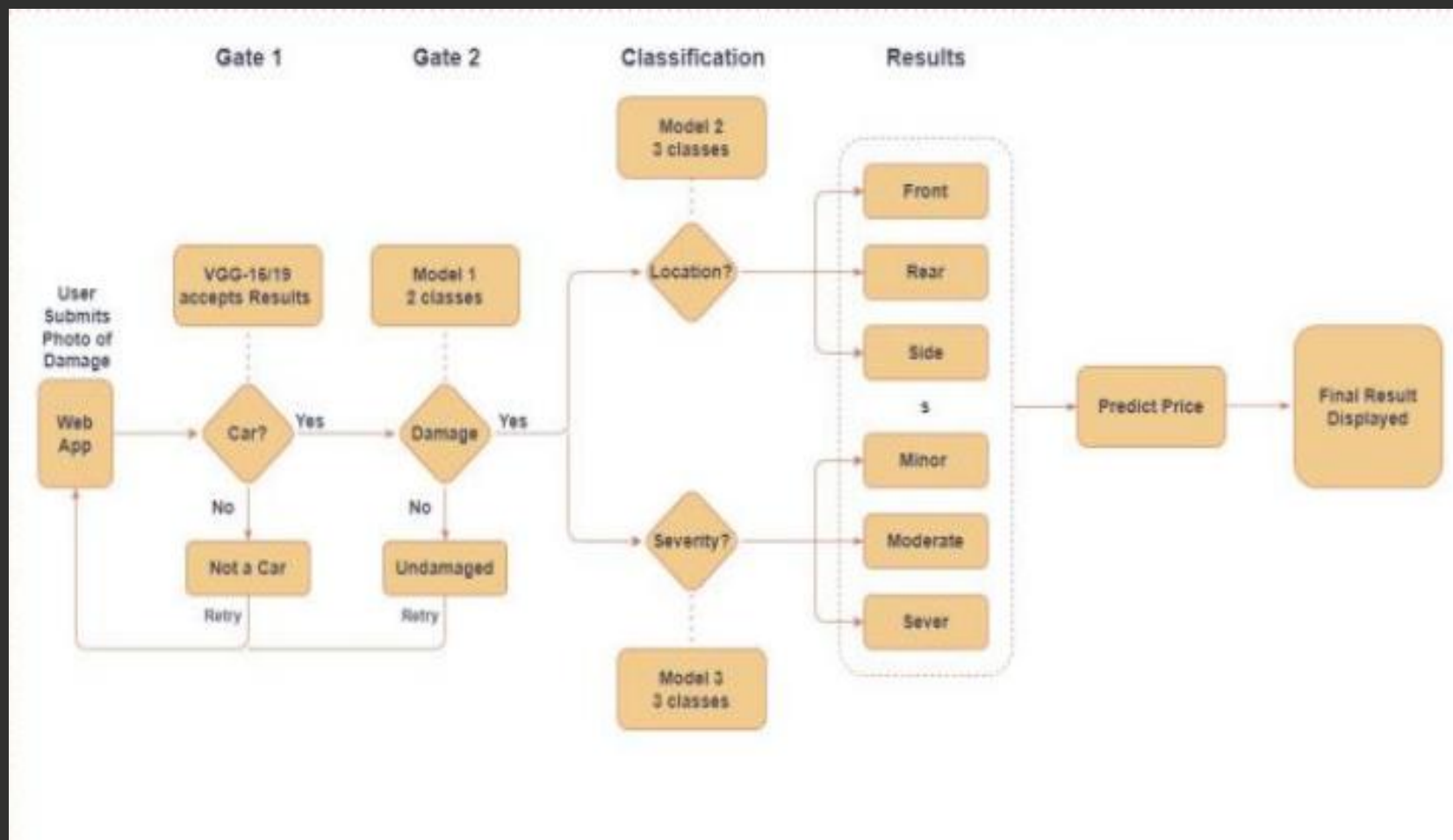
The background is a dark charcoal gray. It features three large, stylized triangles in two shades of teal. One triangle is in the top right corner, pointing downwards. Another is in the bottom left corner, pointing upwards. A third triangle is partially visible on the right edge, also pointing downwards. The number '3' is centered on the left side of the slide.

# 3

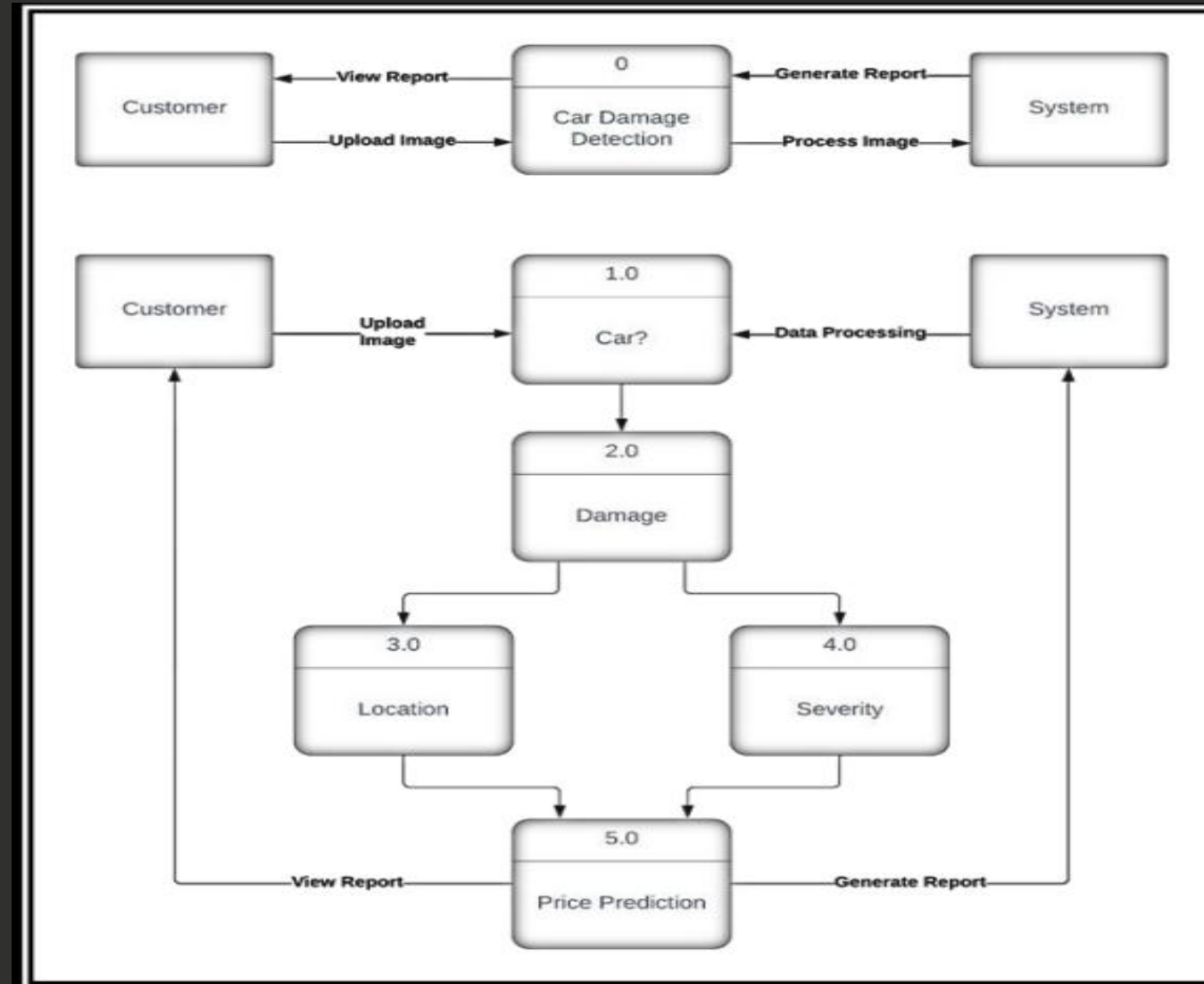
## PART THREE

Architecture, Data Flow  
Diagram, GUI and Experimental  
Result.

# SYSTEM ARCHITECTURE

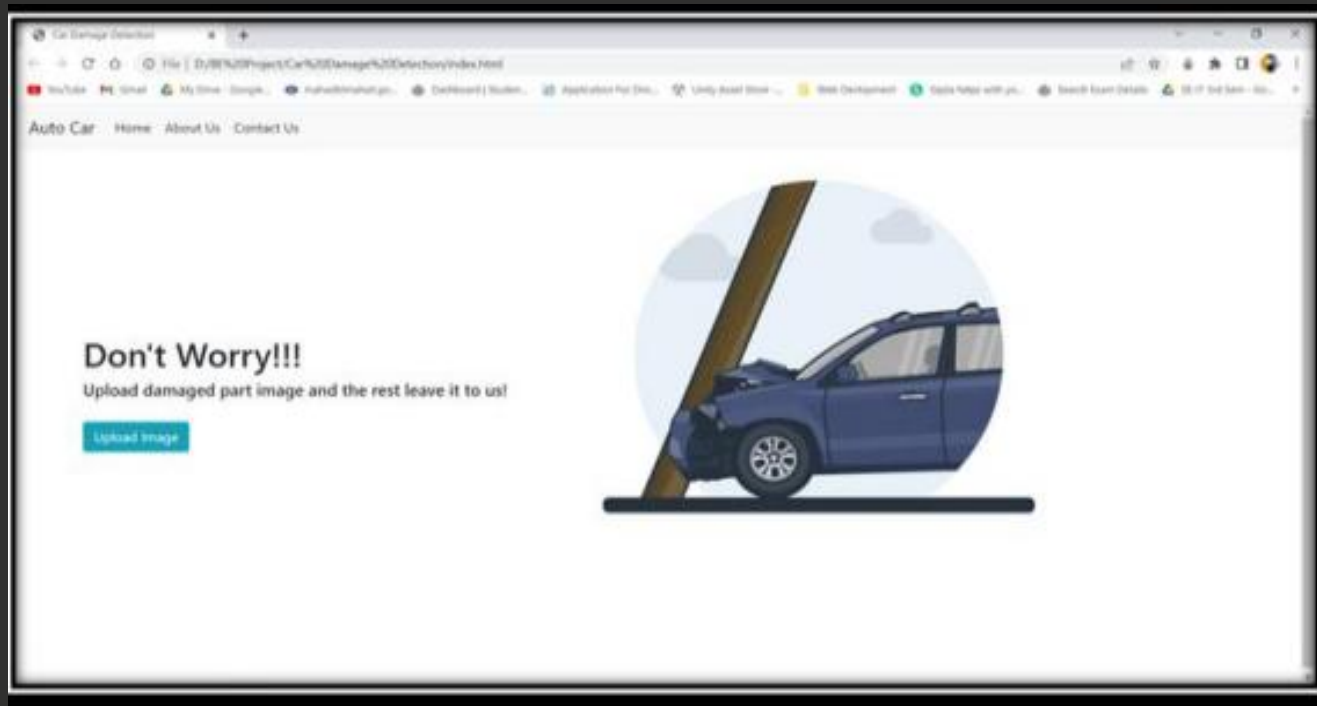


# DATA FLOW DIAGRAM



## GUI AND EXPERIMENTAL RESULTS

Also the first module which detects the car has been integrated using flask. Once the car is detected, it will direct user to next step i.e. Damage Evaluation..



Upload your car's image below:

Choose File cardamage.jpg

Predict Image

Upload your car's image below:

Choose File No file chosen

Predict Image

We we have Successfully detected a car

Proceed to damage prediction

Proceed to Damage Evaluation



# 4

PART Four  
CODE



## Import Libraries

### Code :-

```
from google.colab import  
Drive  
!pip install torch  
import torch
```

### Explination :-

These lines import necessary libraries. google.colab allows mounting your Google Drive, torch is a deep learning library, and we use pip to install it.






## Mount Google Drive

### Code :-

```
drive.mount('/content/drive')
```

### Explanation:-

This line mounts your Google Drive to the /content/drive directory, allowing you to access your files.



## Create Train/Validation Splits

### Code :-

```
trainPath = '/content/drive/MyDrive/Computer Vision CP/dataset/images/train'
valPath = '/content/drive/MyDrive/Computer Vision CP/dataset/images/val'
crsPath = '/content/drive/MyDrive/Computer Vision CP/data/ts'
train_ratio = 0.8
val_ratio = 0.2
# ... (code to count images)
countForTrain = int(len(imgs)*train_ratio)
countForVal = int(len(imgs)*val_ratio)
# ... (code to copy files)shutil.move(crsPath, valPath)
```

### Explanation :-

This section defines paths to your training, validation, and original data folders. It then calculates how many images will go into each folder based on a specified ratio (80% train, 20% validation). Finally, it copies a random selection of images and their corresponding labels to the training and validation folders. The remaining files are moved to the validation folder.

# Train the YOLOv5 Model

## Code :-

```
!python train.py --img 415 --batch 8 --epochs 150 --data dataset.yaml  
--weights  
yolov5s.pt --cache
```

## Explanation :-

This line trains the YOLOv5 object detection model on your data using the train.py script. The parameters specify the image size, batch size, training epochs, data configuration file, pre-trained weights, and caching strategy.

That's 40% of the code explained! The remaining sections focus on using the trained model for detection, visualizing results, and performing image segmentation to potentially refine the detections.



**THANKS**