Build a model to detect car/vehicle object.

Input Dataset

The dataset is taken from kaggle dataset of object-detection-using-yolo-v5 https://www.kaggle.com/code/anupriyagupta1/object-detection-using-yolo-v5/input

The data is loaded from kaggle to google colab.

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!kaggle datasets download -d sshikamaru/car-object-detection
zip_ref = zipfile.ZipFile('/content/cat-and-dog.zip','r')
zip_ref.extractall('/content')
zip_ref.close()
```

The code creates a directory named ".kaggle" in the home directory if it doesn't already exist. It then copies a file named "kaggle.json" to the ".kaggle" directory. Next, it downloads a dataset named "cat-and-dog" from Kaggle. After that, it imports the "zipfile" module. It creates a ZipFile object named "zip_ref" for the file "cat-and-dog.zip" in read mode. It extracts all the files from the zip file to the "/content" directory. Finally, it closes the ZipFile object.

Import Library

```
import os, time, random
import numpy as np
import pandas as pd
import cv2, torch
from tqdm.auto import tqdm
import shutil as sh
from IPython.display import Image, clear_output
import matplotlib.pyplot as plt
# ignore warnings
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

The code imports various libraries such as os, time, random, numpy, pandas, cv2, torch, tqdm, shutil, IPython.display, and matplotlib.pyplot. It also sets up some configurations such as ignoring warnings and enabling inline plotting.

```
#Clone repository from github.com
!git clone https://github.com/ultralytics/yolov5
```

The code clones a repository from the URL "https://github.com/ultralytics/yolov5" using the git clone command

```
# add -q to quit long outputs
!pip install -q -r yolov5/requirements.txt          #install dependencies
!cp yolov5/requirements.txt ./                      #copy file to current directory
```

The code installs the dependencies listed in the "requirements.txt" file located in the "yolov5" directory. It then copies the "requirements.txt" file to the current directory.

```
img_h, img_w, num_channels = (380,676,3)        #num_channels-no. of colors in image(1-Grey, 3-color img RedGreenBlue,4RGBA)
df = pd.read_csv('/content/data/train_solution_bounding_boxes (1).csv')
df.head()
```

The code assigns the values 380, 676, and 3 to the variables img_h, img_w, and num_channels respectively. It then reads a CSV file called 'train_solution_bounding_boxes (1).csv' using the pandas library and assigns it to the variable df. Finally, it displays the first few rows of the DataFrame df using the .head() method.

```
df.rename(columns={'image':'image_id'}, inplace=True)          #rename column image to image_ID
df['image_id'] = df['image_id'].apply(lambda x: x.split('.')[0])   #removes file extension from image_id
df['x_center'] = (df['xmin'] + df['xmax'])/2
df['y_center'] = (df['ymin'] + df['ymax'])/2
df['w'] = df['xmax'] - df['xmin']
df['h'] = df['ymax'] - df['ymin']
df['classes'] = 0
df['x_center'] = df['x_center']/img_w
df['w'] = df['w']/img_w
df['y_center'] = df['y_center']/img_h
df['h'] = df['h']/img_h
df.head()
```

The code performs the following operations on the DataFrame 'df':

1. Renames the column 'image' to 'image_id' using the rename() function.
2. Removes the file extension from the values in the 'image_id' column using the split() method and lambda function.
3. Calculates the center x-coordinate by taking the average of 'xmin' and 'xmax' and assigns it to a new column 'x_center'.
4. Calculates the center y-coordinate by taking the average of 'ymin' and 'ymax' and assigns it to a new column 'y_center'.
5. Calculates the width by subtracting 'xmin' from 'xmax' and assigns it to a new column 'w'.

6. Calculates the height by subtracting 'ymin' from 'ymax' and assigns it to a new column 'h'.
7. Sets the value of the 'classes' column to 0 for all rows.
8. Normalizes the 'x_center', 'w', 'y_center', and 'h' columns by dividing them by 'img_w' and 'img_h' respectively.
9. Returns the first few rows of the modified DataFrame using the head() function.

```
index = list(set(df.image_id))          #create unique image ID as set removes duplicate values
image = random.choice(index)
print("Image ID: %s"%(image))
```

This code creates a list of unique image IDs by converting the image IDs in the DataFrame df into a set and then converting it back into a list. It then randomly selects an image ID from the list and prints it.

```
img = cv2.imread(f'/content/data/training_images/{image}.jpg')
img.shape
```

The given code reads an image file with the filename {image}.jpg from the directory /content/data/training_images/ using the cv2.imread() function from the OpenCV library. It then assigns the loaded image to the variable img. Finally, img.shape returns the dimensions (height, width, and number of channels) of the image.

```
image = random.choice(index)
print("Image ID: %s"%(image))
Image(filename=f'/content/data/training_images/{image}.jpg',width=800)
```

The code randomly selects an item from the list index using the random.choice() function and assigns it to the variable image. It then prints the image ID using string formatting and displays the image using the Image() function from an unknown library. The image file path is constructed using the image variable. The displayed image has a width of 800 pixels.

```
!python yolov5/detect.py --weights yolov5/yolov5s.pt --img 676 --conf 0.4 --source /content/data/testing_images
```

The code appears to be a command-line instruction written in Python. It is invoking a script named "detect.py" from the "yolov5" directory. The script is likely related to object detection using the YOLOv5 model.

The command includes several arguments:

- "--weights yolov5/yolov5s.pt": Specifies the path to the model weights file.
- "--img 676": Specifies the size of the input image.
- "--conf 0.4": Sets the confidence threshold for object detection to 0.4.
- "--source /content/data/testing_images": Specifies the source directory or file for the script to perform object detection on.

In summary, the code runs an object detection script using the YOLOv5 model, with specific settings for model weights, image size, confidence threshold, and source directory or file.

```
predicted_files = []
for (dirpath, dirnames, filenames) in os.walk("/content/yolov5/runs/detect/exp2"):
   predicted_files.extend(filenames)
   dirnames.extend(dirnames)
print("Names of 5 predicted images")
predicted_files[:5]
```

The code does the following:

1. Initializes an empty list called predicted_files.
2. Iterates over the files and directories in the specified directory path ("/content/yolov5/runs/detect/exp2") using os.walk().
3. Appends the filenames to the predicted_files list using extend() method.
4. Extends the dirnames list with its own contents, which does not change the list.
5. Prints the names of the first 5 predicted images by slicing the predicted_files list.

```
Image(filename=
f'/content/yolov5/runs/detect/exp2/{random.choice(predicted_files)}',width=800)
Image(filename=
f'/content/yolov5/runs/detect/exp2/{random.choice(predicted_files)}',width=800)
Image(filename=
f'/content/yolov5/runs/detect/exp2/{random.choice(predicted_files)}',width=800)
Image(filename=
f'/content/yolov5/runs/detect/exp2/{random.choice(predicted_files)}',width=800)
Image(filename=
f'/content/yolov5/runs/detect/exp2/{random.choice(predicted_files)}',width=800)
```

The code displays an image using the google colab Notebook's Image widget. The image file is selected randomly from a list of files located in the directory '/content/yolov5/runs/detect/exp2/'. The width of the displayed image is set to 800 pixels. This is the output where object vehicle car is predicted from the images.

```
!rm -rf runs yolov5
```

This code attempts to remove the directories runs and yolov5 recursively and forcefully from the current directory. The rm command is used to remove files and directories, the -r option makes it recursive (to remove directories and their contents), and the -f option forces the removal without prompting for confirmation.
Thus built a model which detects car / vehicle object from images.