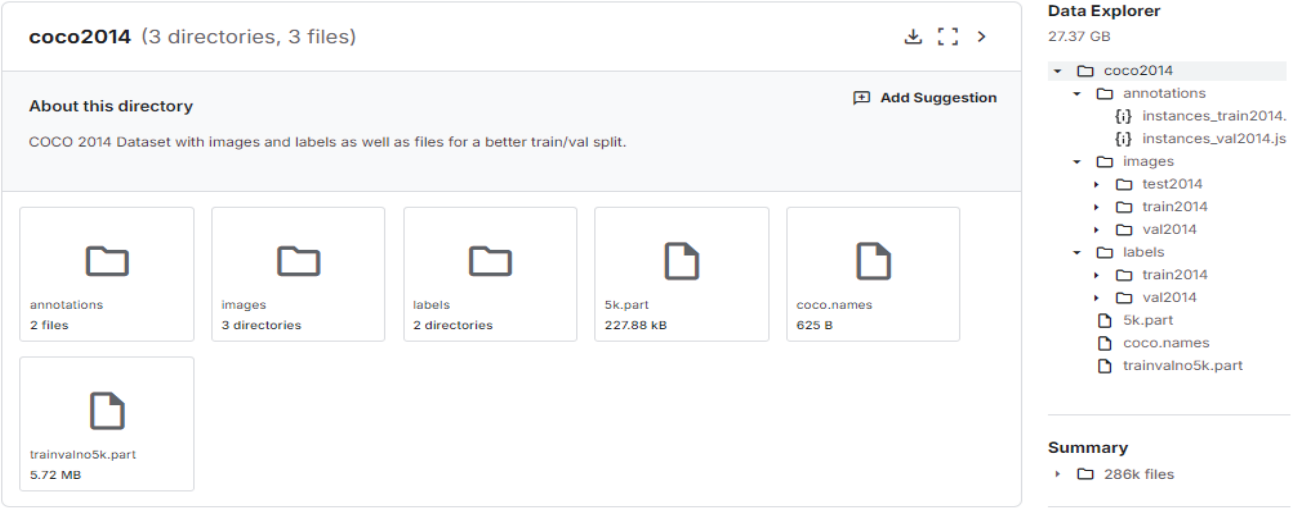
System To Detect And Classify Vehicles In Urban Street Scenes

- **Dataset Name: coco-2014-dataset**



coco2014 (3 directories, 3 files)

About this directory

COCO 2014 Dataset with images and labels as well as files for a better train/val split.

annotations
2 files

images
3 directories

labels
2 directories

5k.part
227.88 kB

coco.names
625 B

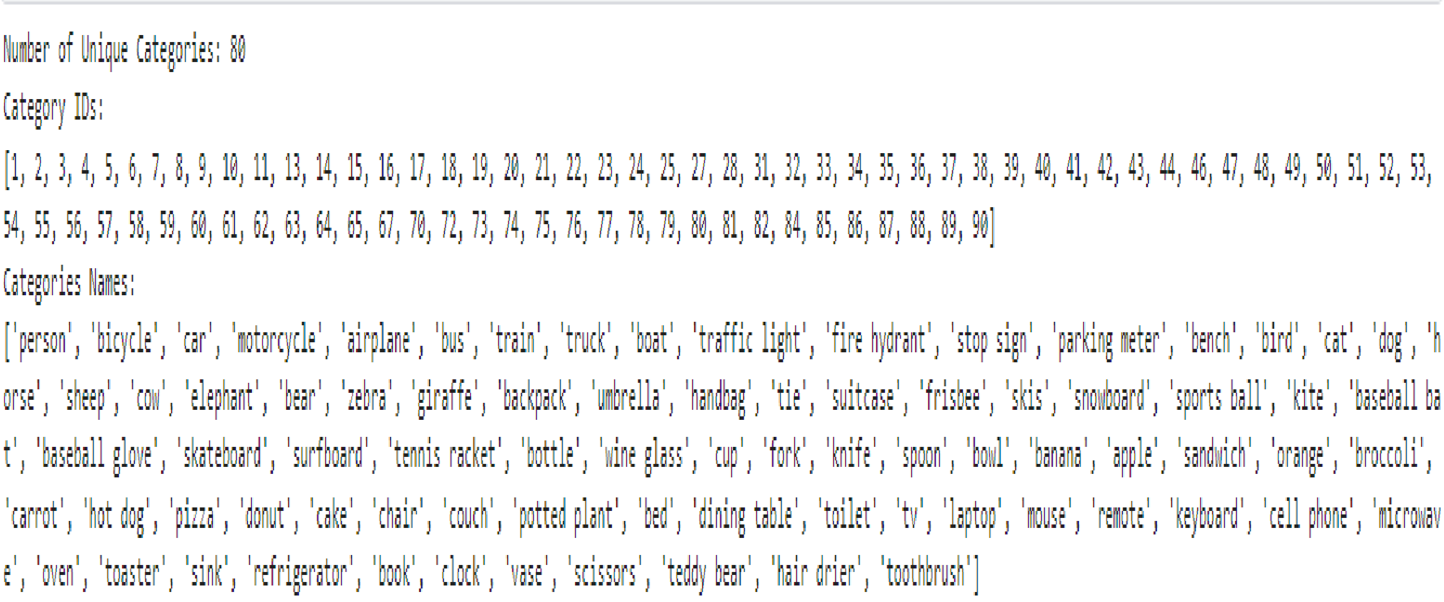
trainvalno5k.part
5.72 MB

Data Explorer
27.37 GB

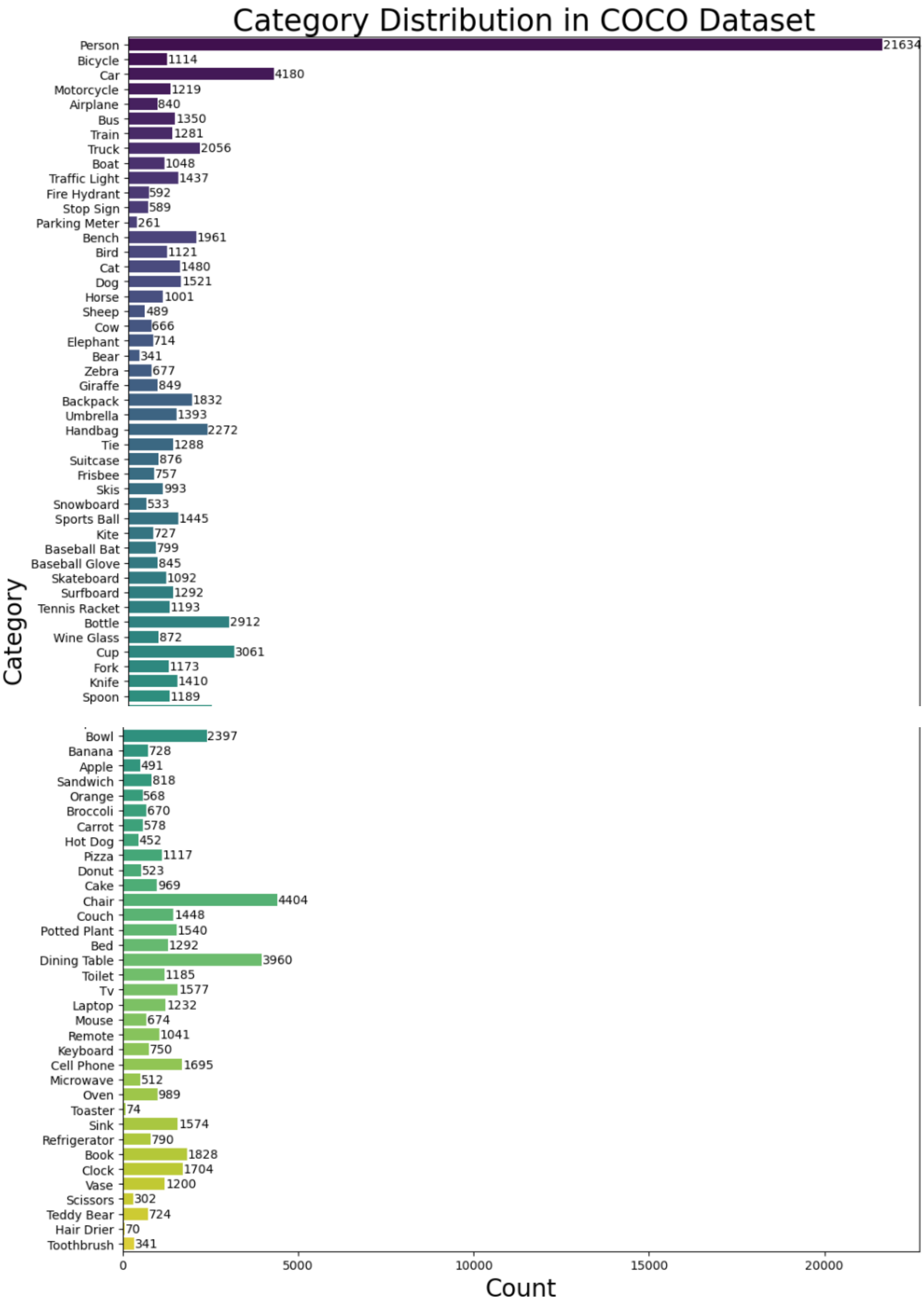
```
coco2014
  annotations
    instances_train2014.
    instances_val2014.js
  images
    test2014
    train2014
    val2014
  labels
    train2014
    val2014
  5k.part
  coco.names
  trainvalno5k.part
```

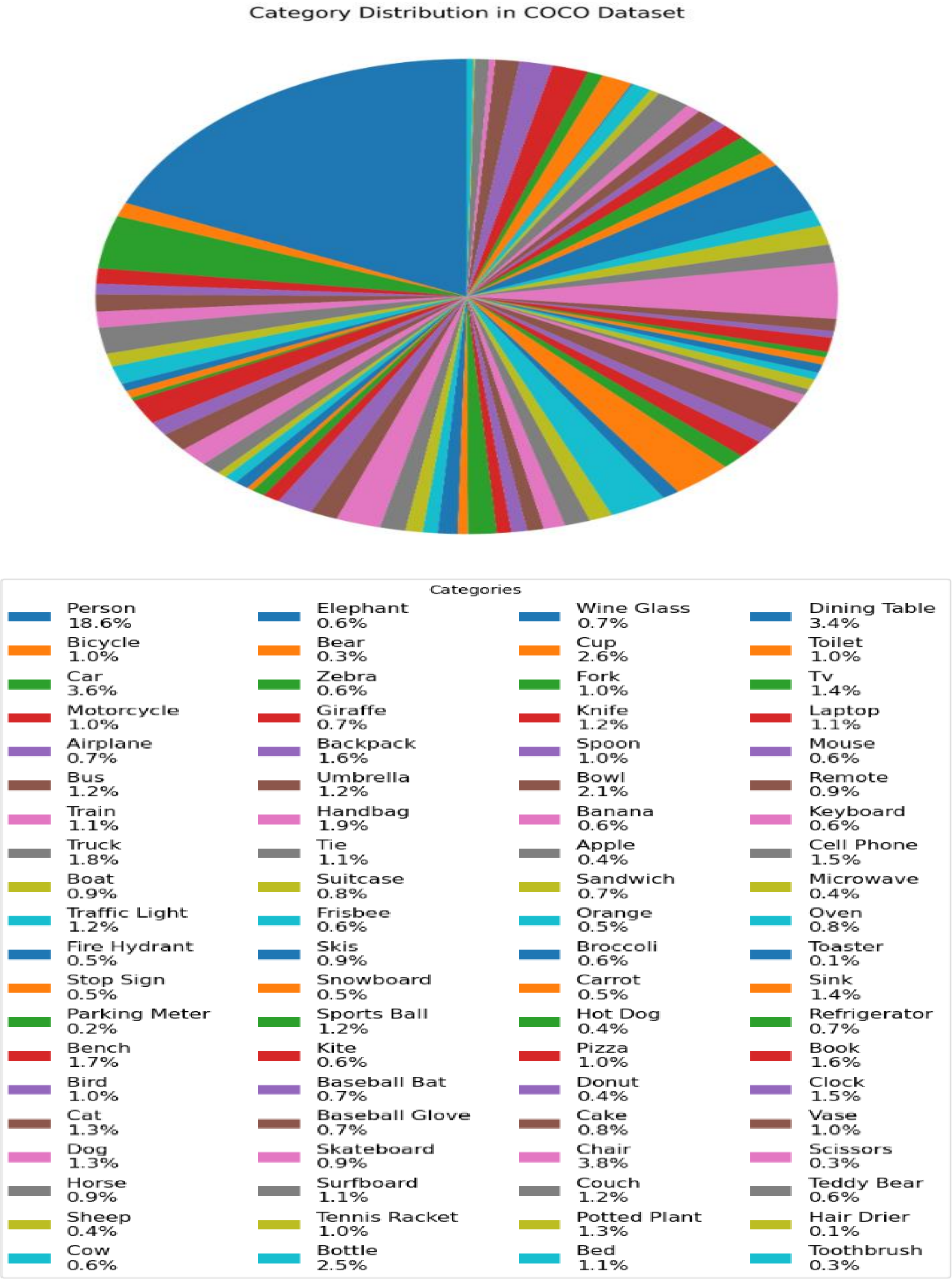
Summary
286k files

- **some visualization related to the dataset:**

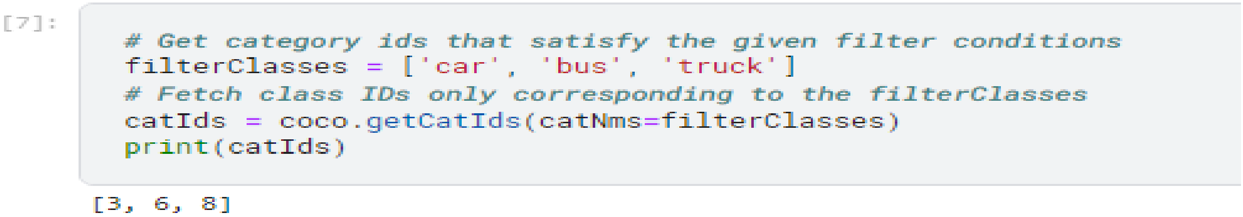


```
Number of Unique Categories: 80
Category IDs:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53,
54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 67, 70, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 84, 85, 86, 87, 88, 89, 90]
Categories Names:
['person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat', 'traffic light', 'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'h
orse', 'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball', 'kite', 'baseball ba
t', 'baseball glove', 'skateboard', 'surfboard', 'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple', 'sandwich', 'orange', 'broccoli',
'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'couch', 'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone', 'microwav
e', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear', 'hair drier', 'toothbrush']
```


Category Distribution in COCO Dataset
Category
Count
Person 21634
Bicycle 1114
Car 4180
Motorcycle 1219
Airplane 840
Bus 1350
Train 1281
Truck 2056
Boat 1048
Traffic Light 1437
Fire Hydrant 592
Stop Sign 589
Parking Meter 261
Bench 1961
Bird 1121
Cat 1480
Dog 1521
Horse 1001
Sheep 489
Cow 666
Elephant 714
Bear 341
Zebra 677
Giraffe 849
Backpack 1832
Umbrella 1393
Handbag 2272
Tie 1288
Suitcase 876
Frisbee 757
Skis 993
Snowboard 533
Sports Ball 1445
Kite 727
Baseball Bat 799
Baseball Glove 845
Skateboard 1092
Surfboard 1292
Tennis Racket 1193
Bottle 2912
Wine Glass 872
Cup 3061
Fork 1173
Knife 1410
Spoon 1189
Bowl 2397
Banana 728
Apple 491
Sandwich 818
Orange 568
Broccoli 670
Carrot 578
Hot Dog 452
Pizza 1117
Donut 523
Cake 969
Chair 4404
Couch 1448
Potted Plant 1540
Bed 1292
Dining Table 3960
Toilet 1185
Tv 1577
Laptop 1232
Mouse 674
Remote 1041
Keyboard 750
Cell Phone 1695
Microwave 512
Oven 989
Toaster 74
Sink 1574
Refrigerator 790
Book 1828
Clock 1704
Vase 1200
Scissors 302
Teddy Bear 724
Hair Drier 70
Toothbrush 341
0
5000
10000
15000
20000


Category Distribution in COCO Dataset
Categories
Person 18.6%
Bicycle 1.0%
Car 3.6%
Motorcycle 1.0%
Airplane 0.7%
Bus 1.2%
Train 1.1%
Truck 1.8%
Boat 0.9%
Traffic Light 1.2%
Fire Hydrant 0.5%
Stop Sign 0.5%
Parking Meter 0.2%
Bench 1.7%
Bird 1.0%
Cat 1.3%
Dog 1.3%
Horse 0.9%
Sheep 0.4%
Cow 0.6%
Elephant 0.6%
Bear 0.3%
Zebra 0.6%
Giraffe 0.7%
Backpack 1.6%
Umbrella 1.2%
Handbag 1.9%
Tie 1.1%
Suitcase 0.8%
Frisbee 0.6%
Skis 0.9%
Snowboard 0.5%
Sports Ball 1.2%
Kite 0.6%
Baseball Bat 0.7%
Baseball Glove 0.7%
Skateboard 0.9%
Surfboard 1.1%
Tennis Racket 1.0%
Bottle 2.5%
Wine Glass 0.7%
Cup 2.6%
Fork 1.0%
Knife 1.2%
Spoon 1.0%
Bowl 2.1%
Banana 0.6%
Apple 0.4%
Sandwich 0.7%
Orange 0.5%
Broccoli 0.6%
Carrot 0.5%
Hot Dog 0.4%
Pizza 1.0%
Donut 0.4%
Cake 0.8%
Chair 3.8%
Couch 1.2%
Potted Plant 1.3%
Bed 1.1%
Dining Table 3.4%
Toilet 1.0%
Tv 1.4%
Laptop 1.1%
Mouse 0.6%
Remote 0.9%
Keyboard 0.6%
Cell Phone 1.5%
Microwave 0.4%
Oven 0.8%
Toaster 0.1%
Sink 1.4%
Refrigerator 0.7%
Book 1.6%
Clock 1.5%
Vase 1.0%
Scissors 0.3%
Teddy Bear 0.6%
Hair Drier 0.1%
Toothbrush 0.3%

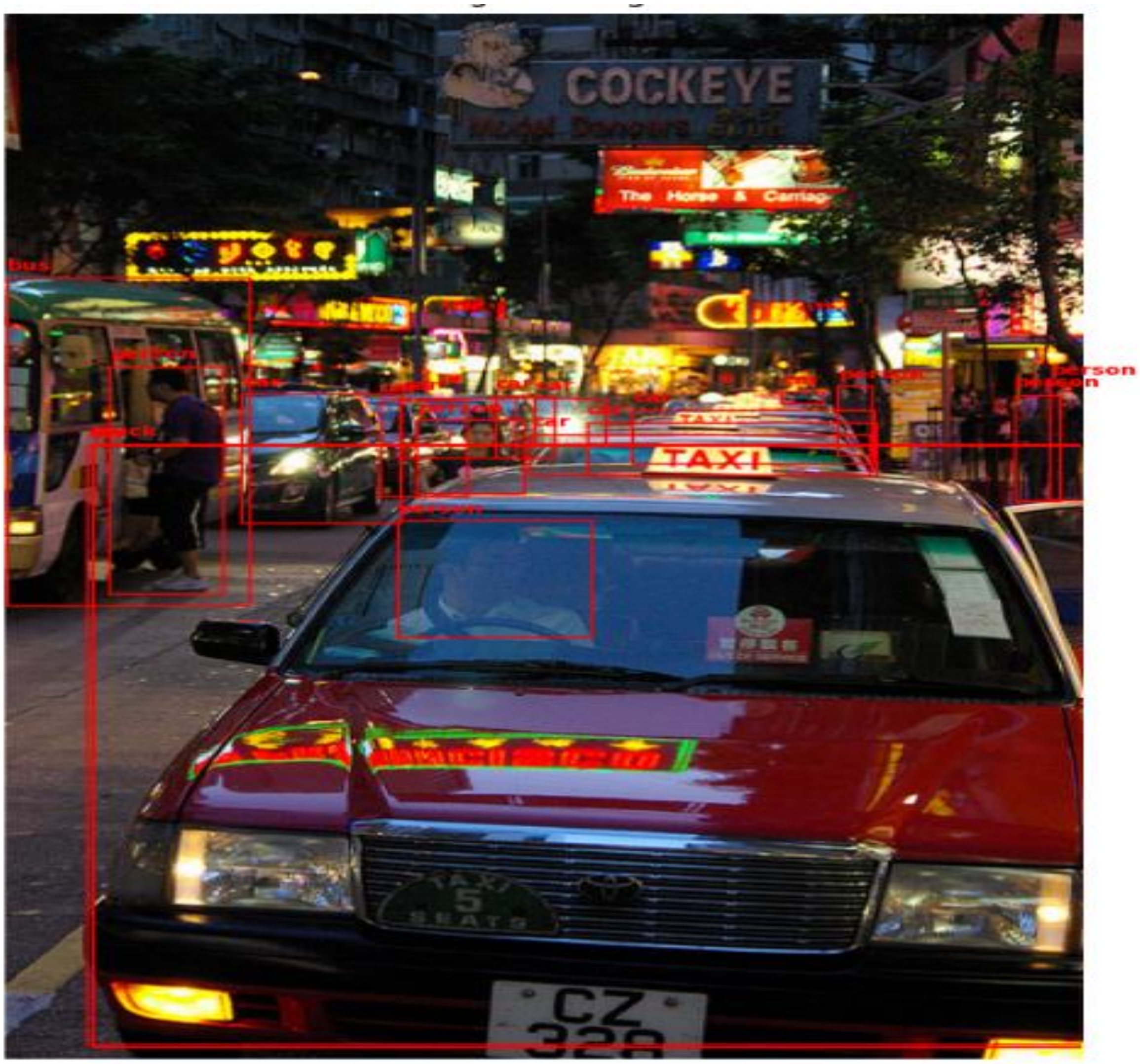
number of categories: 80 which we interested in only 3 as follow:



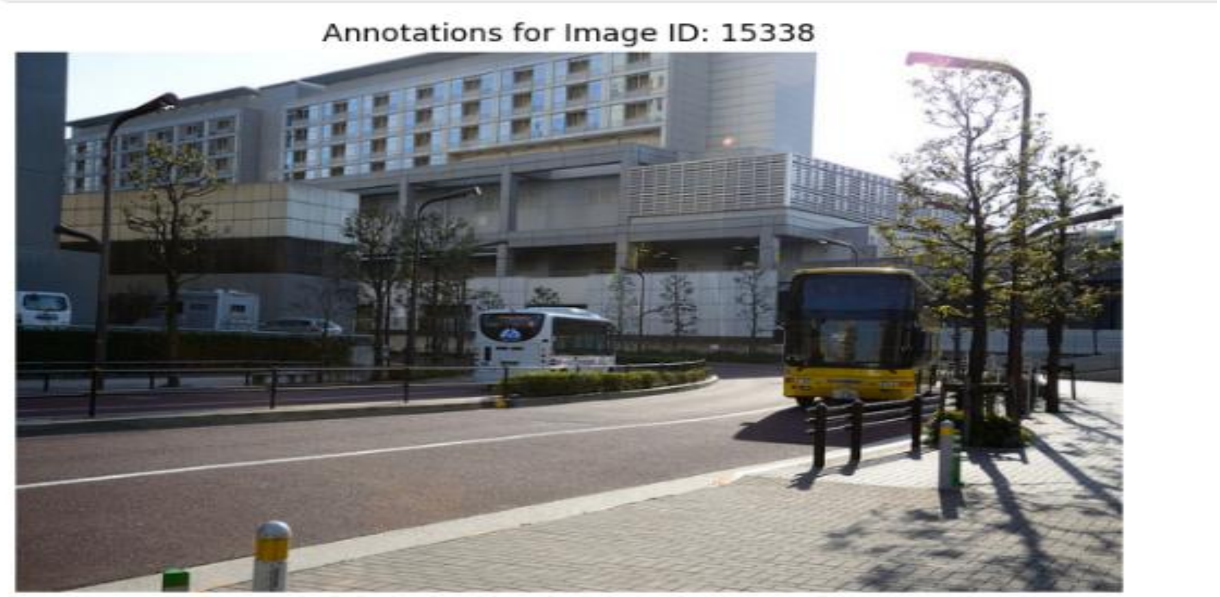
```
[7]:
# Get category ids that satisfy the given filter conditions
filterClasses = ['car', 'bus', 'truck']
# Fetch class IDs only corresponding to the filterClasses
catIds = coco.getCatIds(catNms=filterClasses)
print(catIds)
```

```
[3, 6, 8]
```

- **Custom method Object Detection using Bounding Boxes based on annotations of data (For filtered classes ==>> 'car', 'bus', 'truck')**

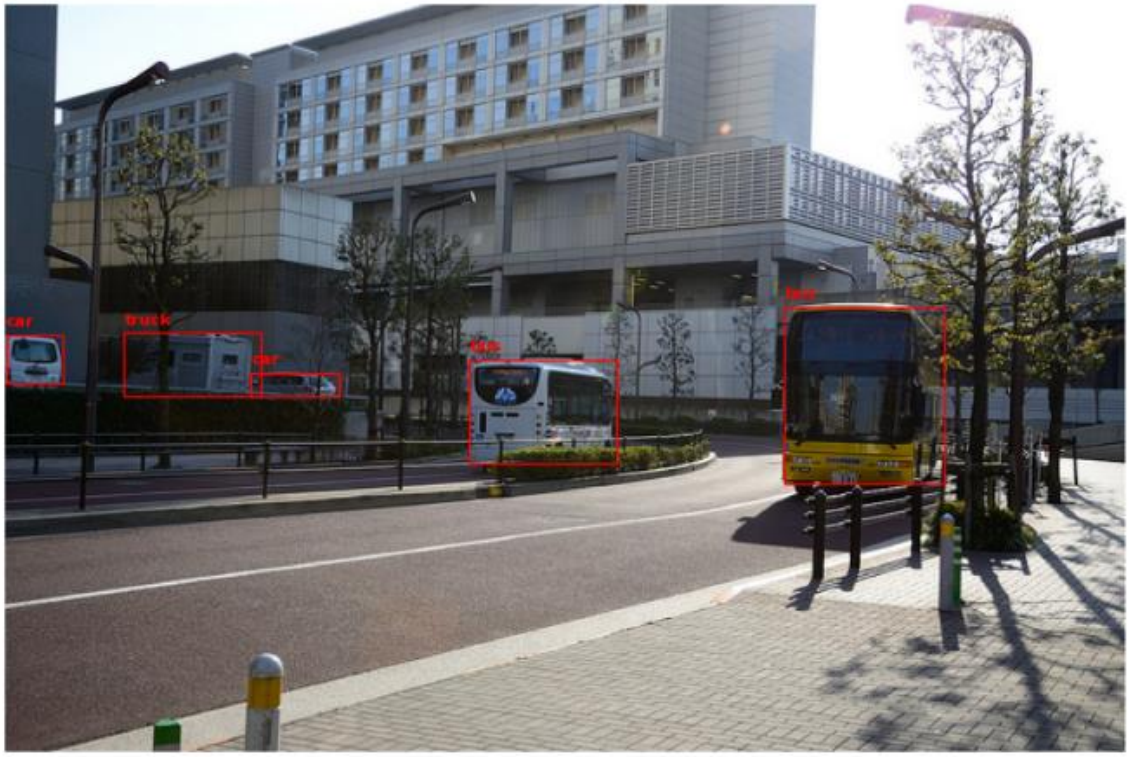


Before:



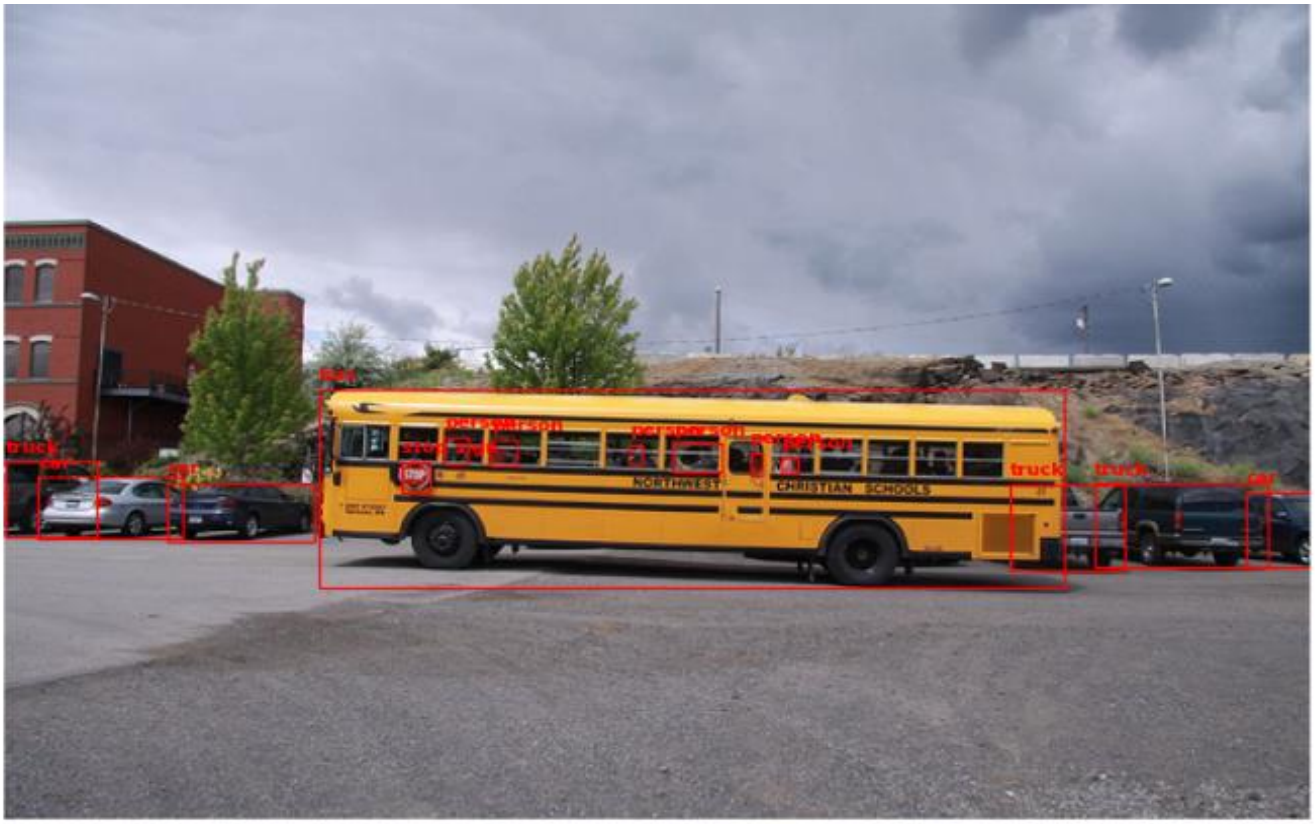
Annotations for Image ID: 15338

After:



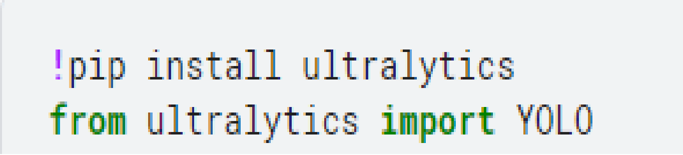
Before

After

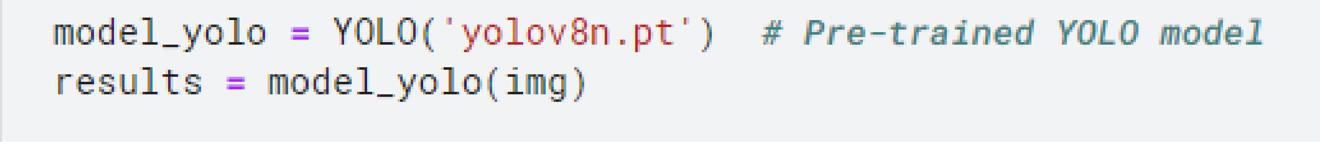


Object Detection: I chose YOLO model version 8 'yolov8n.pt'

- Pre-trained YOLO model doesn’t need to be trained which trained globally in big datasets like official coco datasets
- which is best for real-time performance rather than any other model
- don’t need to download wights just do this command:



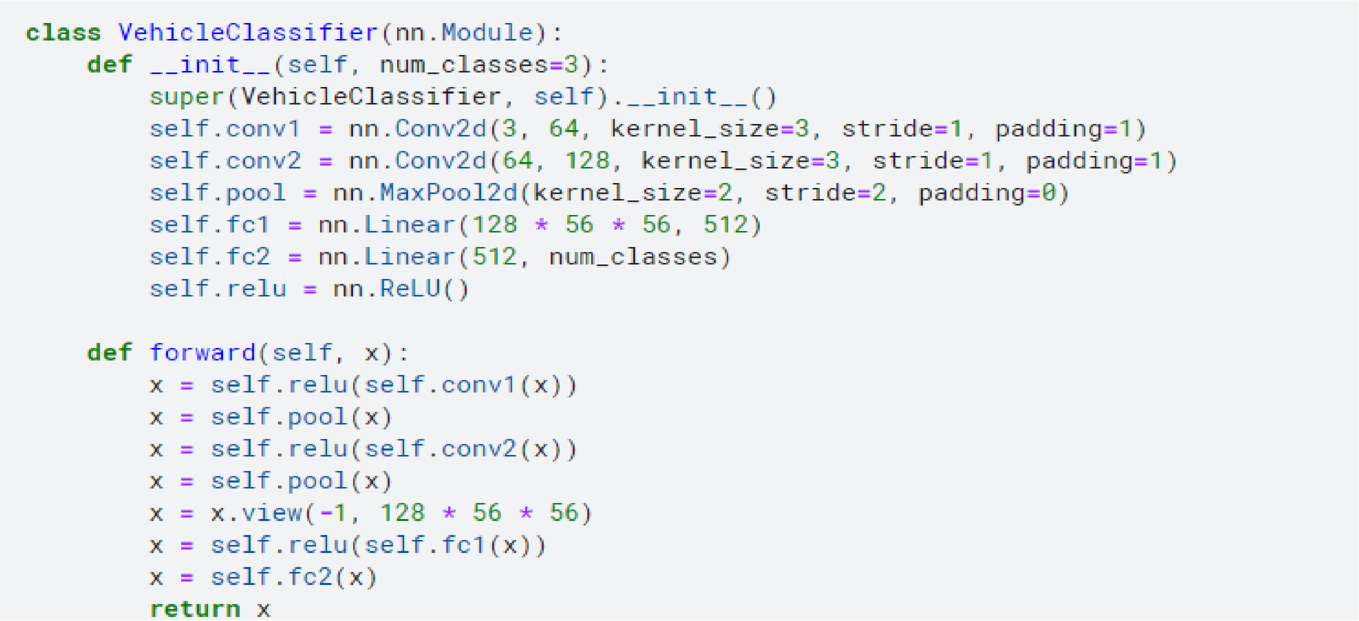
```
!pip install ultralytics
from ultralytics import YOLO
```



```
model_yolo = YOLO('yolov8n.pt')  # Pre-trained YOLO model
results = model_yolo(img)
```

Classification: using Custom CNN

Using Torch vision(nn) & pycocotools Libraries



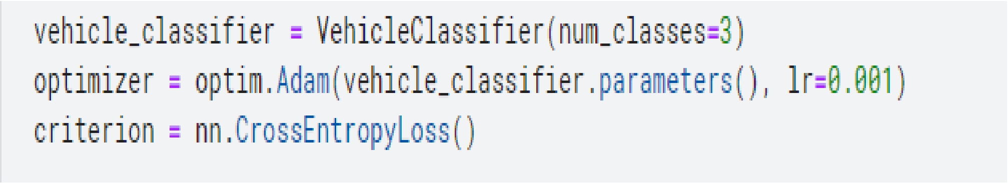
```
class VehicleClassifier(nn.Module):
    def __init__(self, num_classes=3):
        super(VehicleClassifier, self).__init__()
        self.conv1 = nn.Conv2d(3, 64, kernel_size=3, stride=1, padding=1)
        self.conv2 = nn.Conv2d(64, 128, kernel_size=3, stride=1, padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
        self.fc1 = nn.Linear(128 * 56 * 56, 512)
        self.fc2 = nn.Linear(512, num_classes)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.relu(self.conv1(x))
        x = self.pool(x)
        x = self.relu(self.conv2(x))
        x = self.pool(x)
        x = x.view(-1, 128 * 56 * 56)
        x = self.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

Some method of enhancements

series of transformations:

1. **Convert to PIL image**: The image is first converted to a PIL format for easier manipulation.
2. **Resize**: The image is resized to a fixed size (default is 224x224 pixels) to ensure consistency across inputs.
3. **Convert to Tensor**: The image is then converted to a PyTorch tensor for further processing.
4. **Normalize**: The image is normalized using mean and standard deviation values typically used in pre-trained models like ResNet. This step standardizes pixel values to improve model performance.
5. **Add batch dimension**: The image is unsqueezed to add a batch dimension, making it ready for input into a neural network.

• optimization techniques:



```
vehicle_classifier = VehicleClassifier(num_classes=3)
optimizer = optim.Adam(vehicle_classifier.parameters(), lr=0.001)
criterion = nn.CrossEntropyLoss()
```

➢ **optimizer = optim.Adam(vehicle_classifier.parameters(), lr=0.001)**:

The Adam optimizer is used to update the model's weights. It adjusts the parameters based on the gradients computed during backpropagation, with a learning rate of 0.001, balancing convergence speed and stability.

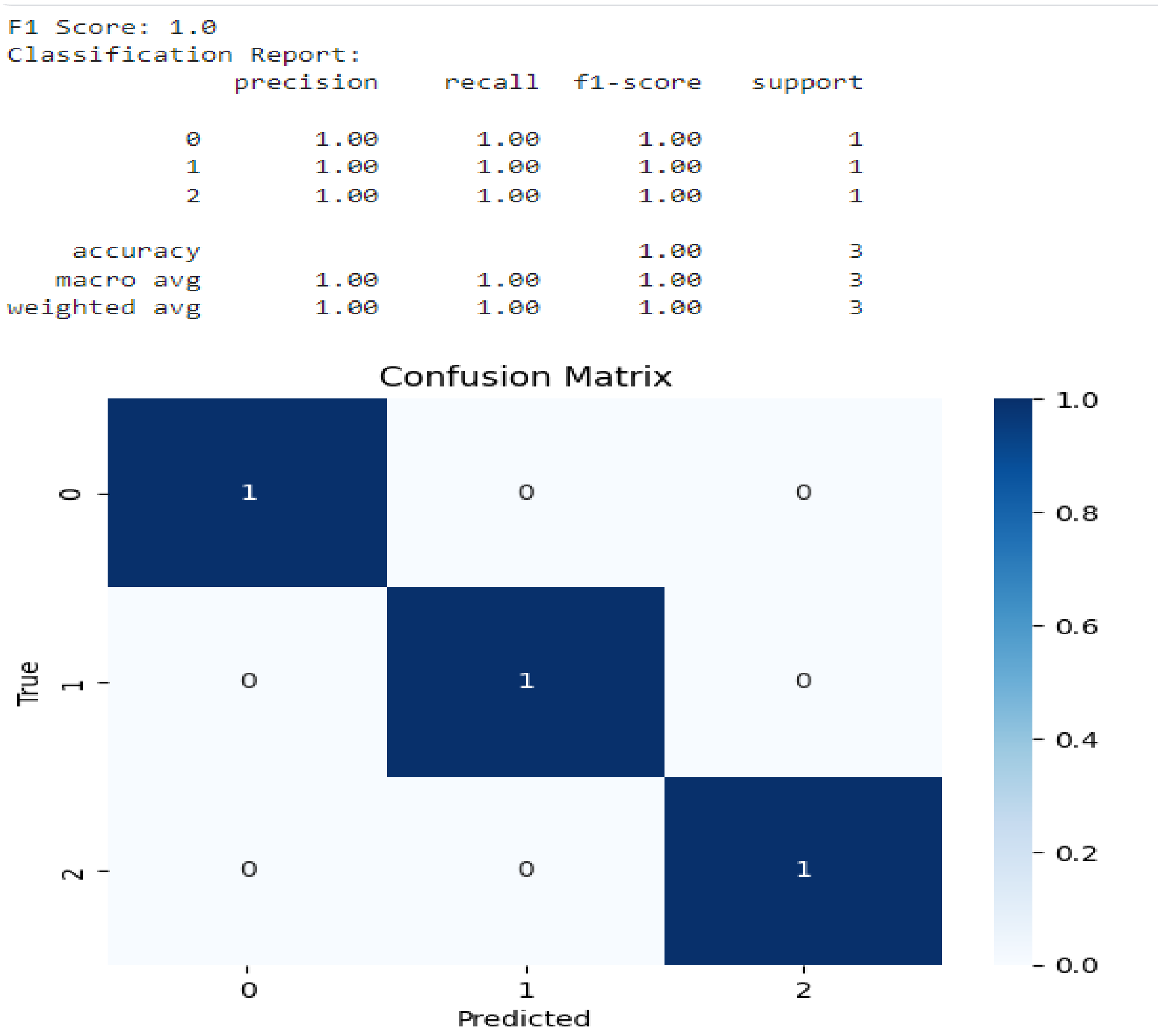
➢ **criterion = nn.CrossEntropyLoss()**:

Cross-entropy loss is chosen as the loss function, commonly used for multi-class classification. It measures the difference between the predicted output probabilities and the true class labels, guiding model optimization.

- **Evaluation:**

(F1 Score and Confusion Matrix)

Classification Report



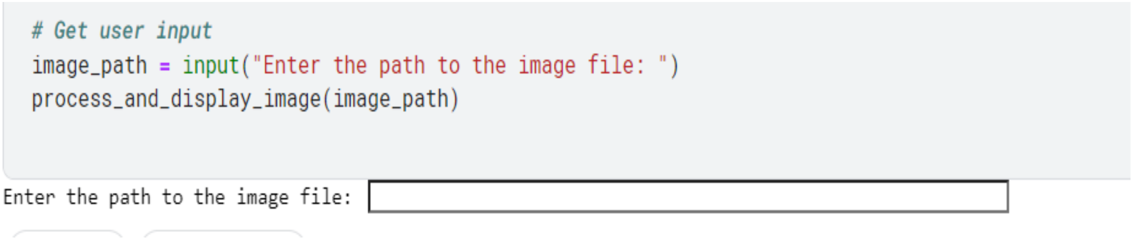
```
F1 Score: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1
           2       1.00      1.00      1.00         1

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3
```

Simple Ui (CLI):

Due to working on Kaggle platform and visibility of dataset online,

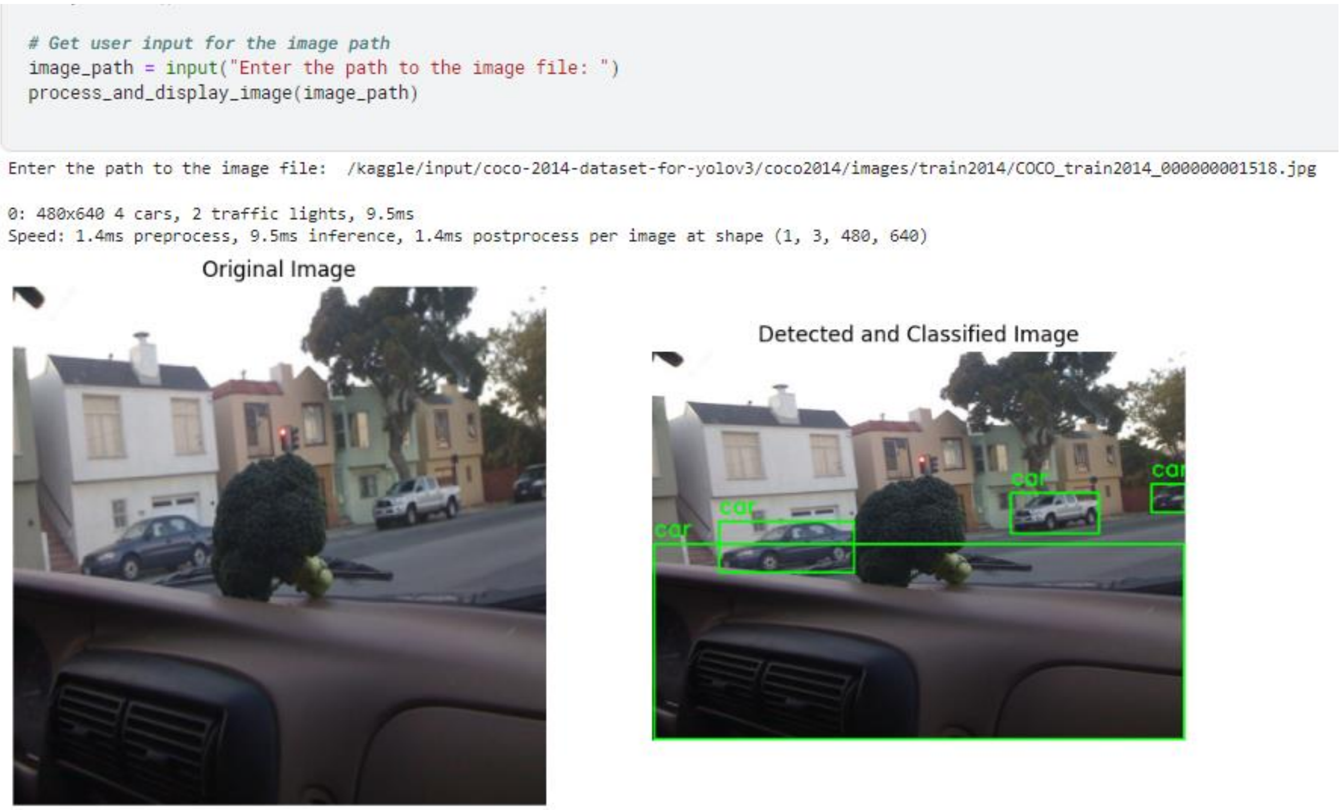
I can’t work use Thinker or any other method of presenting UI which required to be in local pc program (PyCharm or notebook Juber)

Only implement simple ui manage us to put path of image then do the test on it



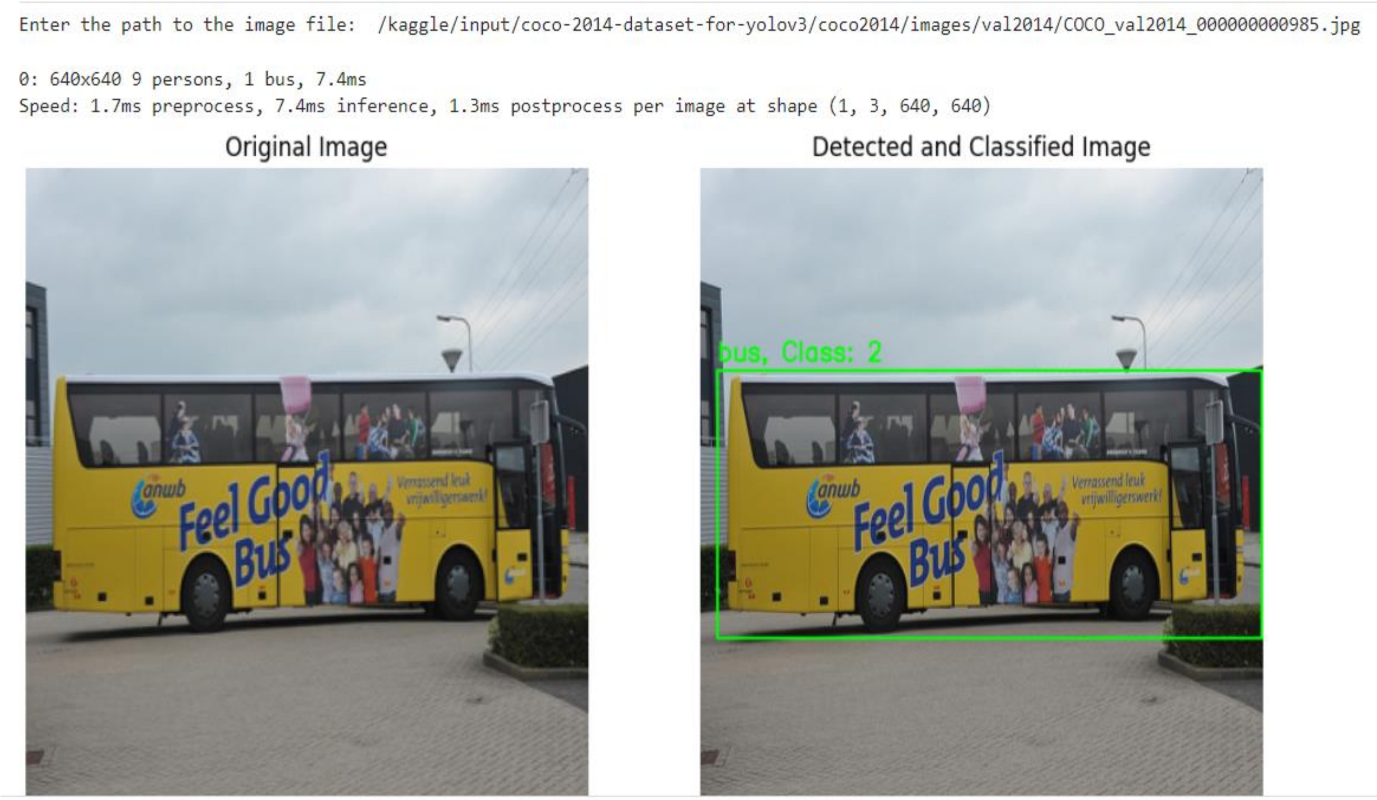
```
# Get user input
image_path = input("Enter the path to the image file: ")
process_and_display_image(image_path)
```

```
Enter the path to the image file: 
```



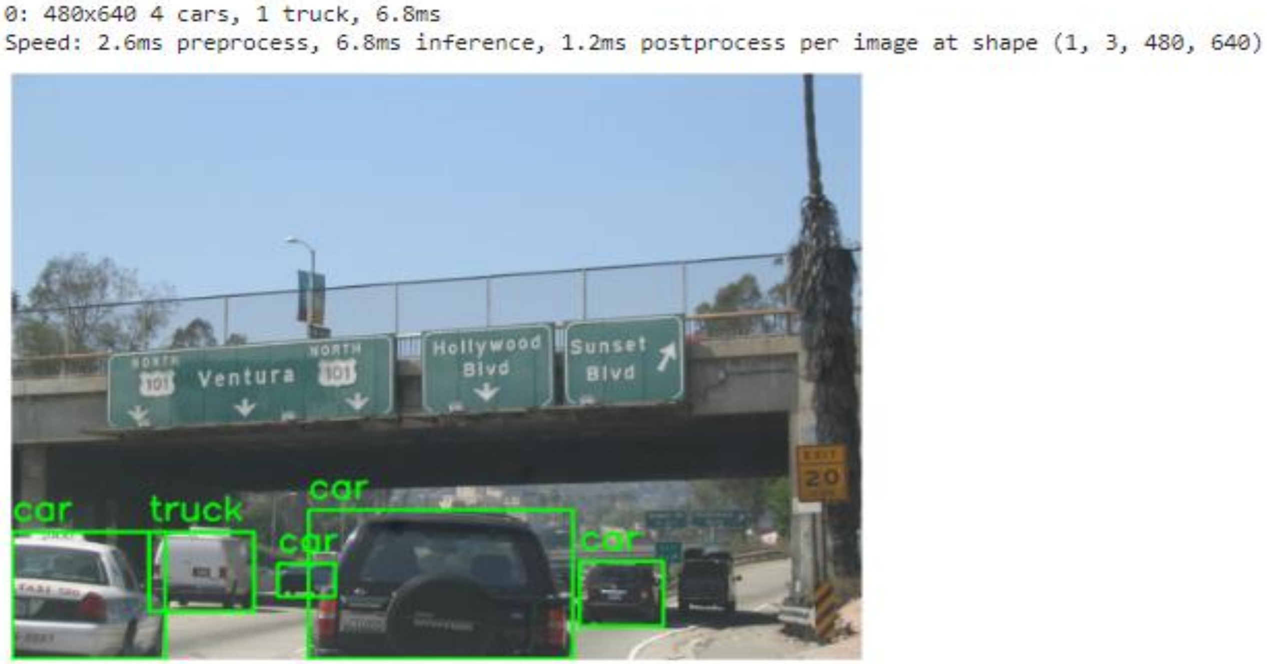
```
# Get user input for the image path
image_path = input("Enter the path to the image file: ")
process_and_display_image(image_path)
```

```
Enter the path to the image file:  /kaggle/input/coco-2014-dataset-for-yolov3/coco2014/images/train2014/COCO_train2014_000000001518.jpg

0: 480x640 4 cars, 2 traffic lights, 9.5ms
Speed: 1.4ms preprocess, 9.5ms inference, 1.4ms postprocess per image at shape (1, 3, 480, 640)
```

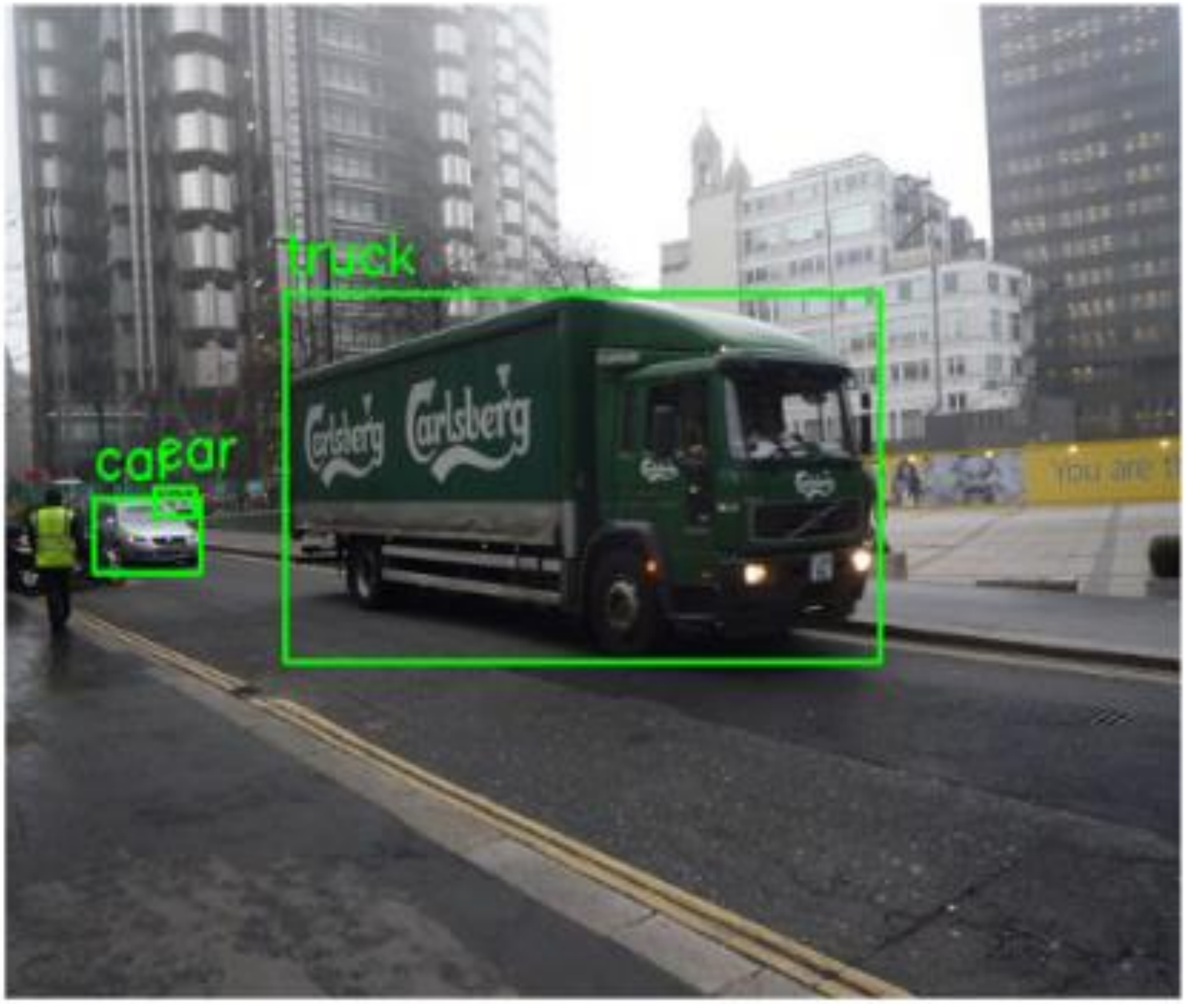


```
Enter the path to the image file:  /kaggle/input/coco-2014-dataset-for-yolov3/coco2014/images/val2014/COCO_val2014_000000000985.jpg

0: 640x640 9 persons, 1 bus, 7.4ms
Speed: 1.7ms preprocess, 7.4ms inference, 1.3ms postprocess per image at shape (1, 3, 640, 640)
```

Some other results:



```
0: 480x640 4 cars, 1 truck, 6.8ms
Speed: 2.6ms preprocess, 6.8ms inference, 1.2ms postprocess per image at shape (1, 3, 480, 640)
```



Thanks 😊