

Object Detection and Autopilot Safety analysis

Part 1 : Object Detection

Import libraries

```
import torch
import os
import shutil
import pathlib
from glob import glob
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Define the data paths

```
image_data_path = None
data_root = None
labels_path = None

def setup_data():
    global image_data_path, data_root, labels_path

    zip_file_path = '/content/drive/MyDrive/DL/od.zip'
    data_root_colab = '/content/drive/MyDrive/DL/Object Detection'
    image_data_path_colab = '/content/drive/MyDrive/DL/Object Detection/Images'
```

```

labels_path_colab = f'{data_root_colab}/labels.csv'

#for Kaggle execution
image_data_path_kaggle = '/kaggle/input/object-detection/Object Detection/Images'
labels_path_kaggle = '/kaggle/input/object-detection/Object Detection/labels.csv'
data_root_kaggle = '.'

if os.path.exists(image_data_path_kaggle):
    image_data_path = image_data_path_kaggle
    data_root = data_root_kaggle
    labels_path = labels_path_kaggle
    print(f'Using Kaggle data ')
else:
    print('Not on kaggle')
    from google.colab import drive
    drive.mount('/content/drive')
    data_root = data_root_colab
    image_data_path = image_data_path_colab
    labels_path = labels_path_colab
    if not os.path.exists(image_data_path):
        from zipfile import ZipFile
        zip_ref = ZipFile(zip_file_path, 'r')
        zip_ref.extractall('/content/drive/MyDrive/DL/')
        zip_ref.close()

    print(f'Using Google Drive data')
    print(f'image data path is {image_data_path}')

    print(f'labels path is at {labels_path}')
setup_data()

```

Using Kaggle data
image data path is /kaggle/input/object-detection/Object Detection/Images
labels path is at /kaggle/input/object-detection/Object Detection/labels.csv

Create the directory structure for yolo training

```

train_label_path = None
val_labels_path = None

```

```
train_image_path = None
val_image_path = None
test_image_path = None

def setup_yolo_paths():
    # create train , validation and test image directories
    global train_label_path
    train_label_path = 'datasets/train/labels'
    global val_labels_path
    val_labels_path = 'datasets/val/labels'
    global train_image_path
    train_image_path = 'datasets/train/images'
    global val_image_path
    val_image_path = 'datasets/val/images'
    global test_image_path
    test_image_path = 'datasets/test/images'

setup_yolo_paths()
```

```
dirs = [train_label_path, val_labels_path, train_image_path, val_image_path, test_image_path]

for d in dirs:
    if os.path.exists(d):
        shutil.rmtree(d)
    os.makedirs(d)
    print(f'created {d}')

if not os.path.exists('datasets/data.yaml'):
    pathlib.Path.touch('datasets/data.yaml')
```

```
created datasets/train/labels
created datasets/val/labels
created datasets/train/images
created datasets/val/images
created datasets/test/images
```

```
%ls datasets
```

```
data.yaml  test/  train/  val/
```

Load the labels csv file

```
import pandas as pd

label_data = pd.read_csv(labels_path, names=['id', 'name', 'x_min', 'y_min', 'x_max', 'y_max'])
label_data = label_data.drop_duplicates('id')
label_data['id'] = label_data['id'].astype(str).str.zfill(8)
label_data = label_data.iloc[0:len(os.listdir(image_data_path))]
label_data['image'] = sorted(os.listdir(image_data_path))
print(label_data.shape)
label_data.head()
```

(5626, 7)

	id	name	x_min	y_min	x_max	y_max	image
0	00000000	pickup_truck	213	34	255	50	00000000.jpg
5	00000001	bus	205	155	568	314	00000001.jpg
11	00000002	motorized_vehicle	334	362	603	479	00000002.jpg
12	00000003	car	480	99	511	120	00000003.jpg
14	00000004	bus	439	32	719	171	00000004.jpg

Estimate Vehicle numbers

```
def copy_image_to_name(row):
    ...
    copies an image to the respective folder
    param : row
    ...
    name = row['name']
    dest = f'{data_root}/image_root/{name}/'
    if os.path.exists(dest):
        shutil.copy(f'{image_data_path}/{row["image"]}', dest)
    else:
```

```
os.makedirs(dest)
shutil.copy(f'{image_data_path}/{row["image"]}', dest)
label_data.apply(copy_image_to_name, axis=1)
```

```
def get_subdirectories(directory) -> list:
    with os.scandir(directory) as entries:
        return [entry.name for entry in entries if entry.is_dir()]

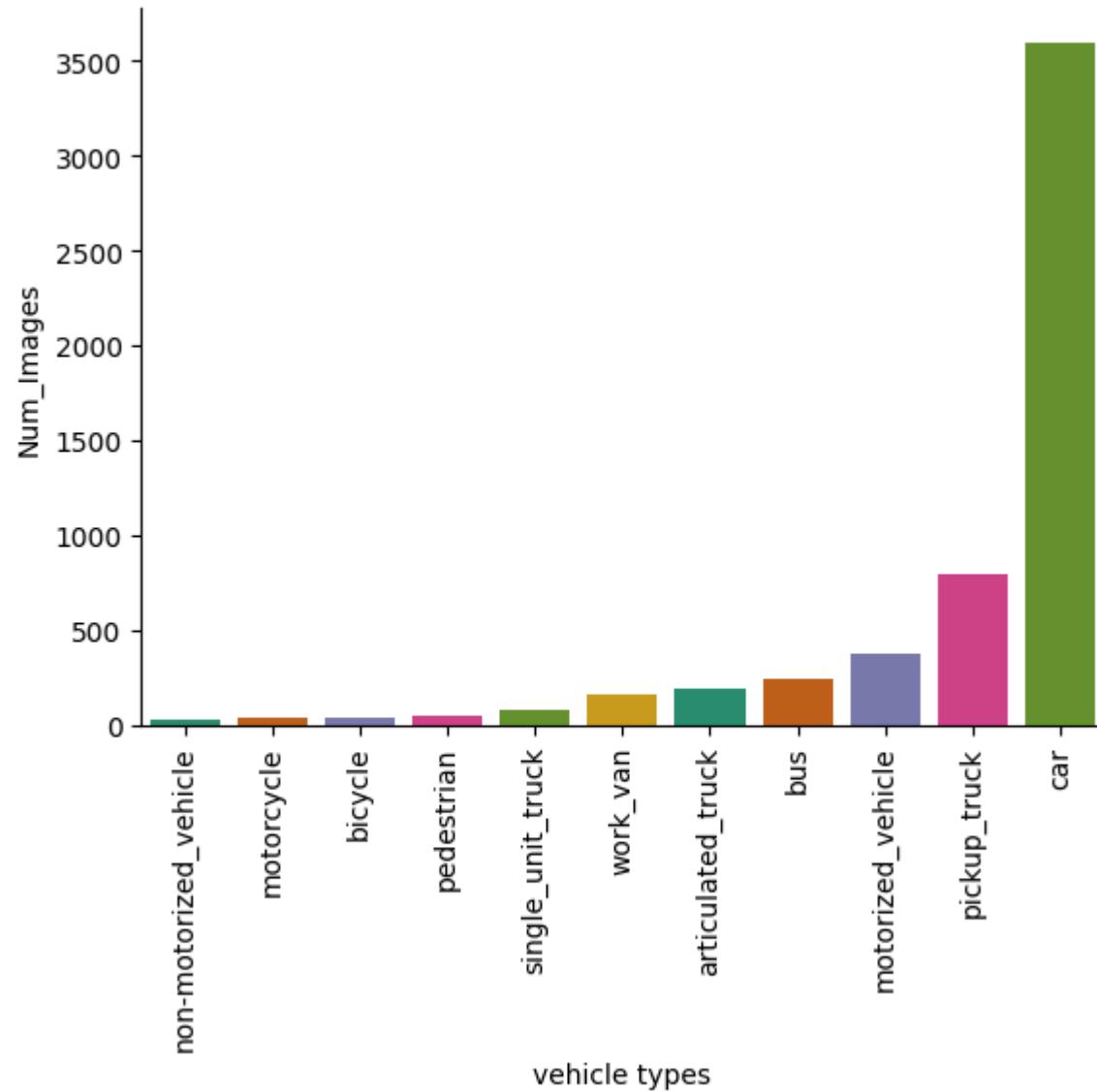
def plot_class_numbers(directory, title) -> pd.DataFrame:
    """
    Plots the number of images in each class
    :param directory: the directory containing the images
    :param title: the title of the plot
    :return: a dataframe containing the number of images in each class
    """
    sub_dirs = get_subdirectories(directory)
    images = []
    image_class = []
    num_images = []
    class_dict = {}

    for d in sub_dirs:
        image_class.append(d)
        num_images.append(len(os.listdir(str(directory) + '/' + d)))

    df = pd.DataFrame({'Image_Class': image_class, 'Num_Images': num_images})
    df = df.sort_values(by='Num_Images', ascending=True).reset_index()

    indices = sorted(df['index'].values.tolist())
    class_names = df['Image_Class'].values
    for i in indices:
        class_dict[i] = class_names[i]
    sns.barplot(data=df, x='Image_Class', y='Num_Images', palette=sns.mpl_palette('Dark2'))
    plt.xlabel('vehicle types')
    plt.title(f' {title} ')
    plt.xticks(rotation=90)
    plt.gca().spines[['top', 'right']].set_visible(False)
    plt.show()
    return df
```

```
plot_class_numbers(f'{data_root}/image_root', '')
```



index		Image_Class	Num_Images
0	8	non-motorized_vehicle	32
1	6	motorcycle	38
2	5	bicycle	43
3	9	pedestrian	56
4	2	single_unit_truck	80
5	10	work_van	161
6	3	articulated_truck	195
7	0	bus	244
8	7	motorized_vehicle	383
9	1	pickup_truck	801
10	4	car	3593

There is substantial class imbalance

Create a map of vehicle types to index

```
import pprint
class_names = label_data.name.unique().tolist()
d = {}
for i in range(len(class_names)):
    d[class_names[i]] = i
pprint.pprint(d)
```

```
{'articulated_truck': 4,
'bicycle': 8,
'bus': 1,
'car': 3,
'motorcycle': 10,
'motorized_vehicle': 2,
'non-motorized_vehicle': 7,
'pedestrian': 6,
'pickup_truck': 0,
'single_unit_truck': 9,
'work_van': 5}
```

Create labels for training and validation

```
import cv2
import numpy as np
import pathlib
def get_label_filename(row):
    return row['id']+'.txt'

def normalize_bb(x_min,y_min,x_max,y_max,id):
    img_path = f'{image_data_path}/{id}.jpg'
    if os.path.exists(img_path):
        img = cv2.imread(img_path)
        image_height,image_width,_ = img.shape
        x_center = (x_min + x_max)/2
        y_center = (y_min + y_max)/2
        width = x_max - x_min
        height = y_max - y_min
        normalized_x_center = x_center / image_width
        normalized_y_center = y_center / image_height
        normalized_width = width / image_width
        normalized_height = height / image_height
        return normalized_x_center, normalized_y_center, normalized_width , normalized_height

def get_label_data(row):
    dims = normalize_bb(row['x_min'], row['y_min'],row['x_max'],row['y_max'],row['id'])
    class_id = d.get(row['name'])
```

```

yolo_annotation = f'{class_id} {dims[0]:.3f} {dims[1]:.3f} {dims[2]:.3f} {dims[0]:.3f}'
return yolo_annotation

def get_image_width(row):
    img_path = f'{image_data_path}/{row['id']}.jpg'
    img = cv2.imread(img_path)
    return img.shape[1]

def get_image_height(row):
    img_path = f'{image_data_path}/{row['id']}.jpg'
    img = cv2.imread(img_path)
    return img.shape[0]

def create_yaml():
    file_path = 'datasets/data.yaml'
    if os.path.exists(file_path):
        os.remove(file_path)

    train = 'train: train'
    val = 'val: val'
    nc = "nc: 11"
    names = f"names: {str(class_names)}"
    lines_to_append = [train, val, nc, names]

    try:
        with open(file_path, 'a') as file:
            for line in lines_to_append:
                file.write(f'{line}\n')
        print('data.yaml created')
    except Exception as e:
        print(f'An error occurred: {e}')

```

Create data.yaml for training

```
create_yaml()
```

```
data.yaml created
```

```
%cat datasets/data.yaml
```

```
train: train
val: val
nc: 11
names: ['pickup_truck', 'bus', 'motorized_vehicle', 'car', 'articulated_truck', 'work_van', 'pedestrian', 'non-motorized_vehicle', 'bicycle', 'single_unit_truck', 'motorcycle']
```

Create annotations in YOLO5 format

```
from glob import glob
import warnings
warnings.filterwarnings('ignore')

num_images = len(glob(f'{image_data_path}/*.jpg'))
num_train_images = int(num_images*0.8)

num_test_images = 60
num_val_images = num_images - num_train_images - num_test_images

print(f'total :{num_images}, train :{num_train_images}, val :{num_val_images}, test:{num_test_images}')

labels_train = label_data.iloc[0:num_train_images]
labels_val = label_data.iloc[num_train_images:num_images-num_test_images]
print(f'train shape : {labels_train.shape}, val shape: {labels_val.shape}'
```

```
total :5626, train : 4500, val :1066, test:60
train shape : (4500, 7), val shape: (1066, 7)
```

```
labels_train['filename'] = labels_train.apply(get_label_filename, axis=1)
labels_train['text'] = labels_train.apply(get_label_data, axis=1)
labels_train['image_width'] = labels_train.apply( get_image_width, axis=1)
labels_train['image_height'] = labels_train.apply( get_image_height, axis=1)

labels_val['filename'] = labels_val.apply( get_label_filename, axis=1)
labels_val['text'] = labels_val.apply( get_label_data, axis=1)
labels_val['image_width'] = labels_val.apply( get_image_width, axis=1)
labels_val['image_height'] = labels_val.apply( get_image_height, axis=1)
```

```
print(f'train shape : {labels_train.shape}, val shape: {labels_val.shape}')
labels_train.sample(10)
```

train shape : (4500, 11), val shape: (1066, 11)

	id	name	x_min	y_min	x_max	y_max	image	filename	text	image_width	image_height
10077	00003219	car	0	60	67	95	00003219.jpg	00003219.txt	3 0.098 0.340 0.196 0.098	342	228
1049	00000326	bus	136	49	455	212	00000326.jpg	00000326.txt	1 0.410 0.272 0.443 0.410	720	480
6646	00002095	pickup_truck	136	128	540	304	00002095.jpg	00002095.txt	0 0.469 0.450 0.561 0.469	720	480
13820	00004413	car	486	34	512	58	00004413.jpg	00004413.txt	3 0.693 0.096 0.036 0.693	720	480
1855	00000574	bus	34	120	301	289	00000574.jpg	00000574.txt	1 0.233 0.426 0.371 0.233	720	480
2623	00000843	motorized_vehicle	313	34	349	52	00000843.jpg	00000843.txt	2 0.460 0.090 0.050 0.460	720	480
11830	00003779	car	575	103	635	128	00003779.jpg	00003779.txt	3 0.840 0.241 0.083 0.840	720	480
9631	00003054	car	72	134	124	168	00003054.jpg	00003054.txt	3 0.136 0.315 0.072 0.136	720	480
11390	00003625	car	202	304	339	398	00003625.jpg	00003625.txt	3 0.376 0.731 0.190 0.376	720	480
7300	00002285	car	610	169	653	212	00002285.jpg	00002285.txt	3 0.877 0.397 0.060 0.877	720	480

Define function to create the yolo label file

Create label files for train and validation images

```
from pathlib import Path
# create the label file
def create_label_file(label_path,file_name,text):
    f = Path(f'{label_path}/{file_name}')
    f.write_text(text)

labels_train.apply(lambda row: create_label_file(train_label_path,row['filename'],row['text']) , axis =1)
_labels_val.apply(lambda row: create_label_file(val_labels_path,row['filename'],row['text']) , axis =1)
```

Copy the images to train, val and test folders

```
# copy train images
from tqdm import tqdm

def _copy(src,dest,file_name):
    if not src.startswith('.'):
        src_file = f'{src}/{file_name}'
        if not os.path.isdir(src_file):
            shutil.copy(src_file,dest)

num_train_images = labels_train.shape[0]
num_val_images = labels_val.shape[0]

images = sorted(os.listdir(image_data_path))

print(num_train_images,num_train_images+num_val_images,num_test_images)
print('Copying training images')
for i in tqdm(range(0,num_train_images)):
    _copy(image_data_path,f'{train_image_path}/',images[i])

print('Copying validation images')
for i in tqdm(range(num_train_images,num_train_images+num_val_images)):
    _copy(image_data_path,f'{val_image_path}/',images[i])

print('copying test imges')
for i in tqdm(range(num_train_images+num_val_images,num_images)):
    _copy(image_data_path,f'{test_image_path}/',images[i])
```

```
4500 5566 60
Copying training images
100%|██████████| 4500/4500 [00:11<00:00, 397.92it/s]
Copying validation images
100%|██████████| 1066/1066 [00:02<00:00, 410.30it/s]
copying test imges
100%|██████████| 60/60 [00:00<00:00, 436.67it/s]
```

install ultralytics

```
!pip install ultralytics > install.txt
```

ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.

pylibcugraph-cu12 24.12.0 requires pylibraft-cu12==24.12.*, but you have pylibraft-cu12 25.2.0 which is incompatible.

pylibcugraph-cu12 24.12.0 requires rmm-cu12==24.12.*, but you have rmm-cu12 25.2.0 which is incompatible.

Train the model

```
# The kernel restarts once when this code is run. Reimporting Libraries again
import torch
import os
import shutil
import pathlib
from glob import glob
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from ultralytics import YOLO

data_yaml_kaggle = "/kaggle/working/datasets/data.yaml"
data_yaml_colab = "/content/datasets/data.yaml"
epochs = 100
patience = 10
data_yaml = data_yaml_kaggle if os.path.exists(data_yaml_kaggle) else data_yaml_colab
model = YOLO("yolov5su.pt")
train_results=model.train(data=data_yaml, epochs=epochs, patience = patience, imgsz=640)
```

Ultralytics 8.3.107 🚀 Python-3.11.11 torch-2.5.1+cu124 CUDA:0 (Tesla T4, 15095MiB)

engine/trainer: task=detect, mode=train, model=yolov5su.pt, data=/kaggle/working/datasets/data.yaml, epochs=100, time=None, patience=10, batch=16, imgsz=640, save=True, save_period=-1, cache=False, device=None, workers=8, project=None, name=train3, exist_ok=False, pretrain ed=True, optimizer=auto, verbose=True, seed=0, deterministic=True, single_cls=False, rect=False, cos_lr=False, close_mosaic=10, resume=False, amp=True, fraction=1.0, profile=False, freeze=None, multi_scale=False, overlap_mask=True, mask_ratio=4, dropout=0.0, val=True, split=val, save_json=False, conf=None, iou=0.7, max_det=300, half=False, dnn=False, plots=True, source=None, vid_stride=1, stream_buffer=False, visualize=False, augment=False, agnostic_nms=False, classes=None, retina_masks=False, embed=None, show=False, save_frames=False, save_txt=False, save_conf=False, save_crop=False, show_labels=True, show_conf=True, show_boxes=True, line_width=None, format=torchscript, keras=False, optimize=False, int8=False, dynamic=False, simplify=True, opset=None, workspace=None, nms=False, lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=7.5, cls=0.5, df=1.5, pose=12.0, kobj=1.0, nbs=64, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, bgr=0.0, mosaic=1.0, mixup=0.0, copy_paste=0.0, copy_paste_mode=flip, auto_augment=randaugment, erasing=0.4, crop_fraction=1.0, cfg=None, tracker=botsort.yaml, save_dir=runs/detect/train3

Overriding model.yaml nc=80 with nc=11

	from	n	params	module	arguments
0	-1	1	3520	ultralytics.nn.modules.conv.Conv	[3, 32, 6, 2, 2]
1	-1	1	18560	ultralytics.nn.modules.conv.Conv	[32, 64, 3, 2]
2	-1	1	18816	ultralytics.nn.modules.block.C3	[64, 64, 1]
3	-1	1	73984	ultralytics.nn.modules.conv.Conv	[64, 128, 3, 2]
4	-1	2	115712	ultralytics.nn.modules.block.C3	[128, 128, 2]
5	-1	1	295424	ultralytics.nn.modules.conv.Conv	[128, 256, 3, 2]
6	-1	3	625152	ultralytics.nn.modules.block.C3	[256, 256, 3]
7	-1	1	1180672	ultralytics.nn.modules.conv.Conv	[256, 512, 3, 2]
8	-1	1	1182720	ultralytics.nn.modules.block.C3	[512, 512, 1]
9	-1	1	656896	ultralytics.nn.modules.block.SPPF	[512, 512, 5]
10	-1	1	131584	ultralytics.nn.modules.conv.Conv	[512, 256, 1, 1]
11	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
12	[-1, 6]	1	0	ultralytics.nn.modules.conv.Concat	[1]
13	-1	1	361984	ultralytics.nn.modules.block.C3	[512, 256, 1, False]
14	-1	1	33024	ultralytics.nn.modules.conv.Conv	[256, 128, 1, 1]
15	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
16	[-1, 4]	1	0	ultralytics.nn.modules.conv.Concat	[1]
17	-1	1	90880	ultralytics.nn.modules.block.C3	[256, 128, 1, False]
18	-1	1	147712	ultralytics.nn.modules.conv.Conv	[128, 128, 3, 2]
19	[-1, 14]	1	0	ultralytics.nn.modules.conv.Concat	[1]
20	-1	1	296448	ultralytics.nn.modules.block.C3	[256, 256, 1, False]
21	-1	1	590336	ultralytics.nn.modules.conv.Conv	[256, 256, 3, 2]
22	[-1, 10]	1	0	ultralytics.nn.modules.conv.Concat	[1]
23	-1	1	1182720	ultralytics.nn.modules.block.C3	[512, 512, 1, False]
24	[17, 20, 23]	1	2120305	ultralytics.nn.modules.head.Detect	[11, [128, 256, 512]]

YOLOv5s summary: 153 layers, 9,126,449 parameters, 9,126,433 gradients, 24.1 GFLOPs

Transferred 421/427 items from pretrained weights

TensorBoard: Start with 'tensorboard --logdir runs/detect/train3', view at <http://localhost:6006/>

Freezing layer 'model.24.dfl.conv.weight'

AMP: running Automatic Mixed Precision (AMP) checks...

AMP: checks passed ✓

train: Scanning /kaggle/working/datasets/train/labels.cache... 4500 images, 0 backgrounds, 0 corrupt: 100%|██████████| 4500/4500 [00:00 <?, ?it/s]

albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01, num_output_channels=3, method='weighted_average'), CLAHE(p=0.01, clip_limit=(1.0, 4.0), tile_grid_size=(8, 8))

val: Scanning /kaggle/working/datasets/val/labels.cache... 1066 images, 0 backgrounds, 0 corrupt: 100%|██████████| 1066/1066 [00:00 <?, ?it/s]

Plotting labels to runs/detect/train3/labels.jpg...

optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and 'momentum=0.937' and determining best 'optimizer', 'lr0' and 'momentum' automatically...

optimizer: AdamW(lr=0.000667, momentum=0.9) with parameter groups 69 weight(decay=0.0), 76 weight(decay=0.0005), 75 bias(decay=0.0)

TensorBoard: model graph visualization added ✓

Image sizes 640 train, 640 val

Using 2 dataloader workers

Logging results to **runs/detect/train3**

Starting training for 100 epochs...

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
1/100	4.29G	1.887	4.267	1.829	12	640: 100% ██████████ 282/282 [01:11<00:00, 3.96it/s]	
	Class	Images	Instances	Box(P)	R	mAP50	mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.20it/s]
2/100	4.53G	1.631	2.714	1.666	11	640: 100% ██████████ 282/282 [01:10<00:00, 4.01it/s]	
	Class	Images	Instances	Box(P)	R	mAP50	mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.19it/s]
3/100	4.53G	1.605	2.582	1.647	11	640: 100% ██████████ 282/282 [01:11<00:00, 3.95it/s]	
	Class	Images	Instances	Box(P)	R	mAP50	mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.18it/s]

		all	1066	1066	0.606	0.0992	0.0592	0.0328
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size		
4/100	4.53G Class	1.548 Images	2.476 Instances	1.614 Box(P)	9 R	640: 100% ██████████ 282/282 [01:10<00:00, 3.97it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.15it/		
s]		all	1066	1066	0.522	0.136	0.0906	0.051
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size		
5/100	4.53G Class	1.481 Images	2.368 Instances	1.554 Box(P)	9 R	640: 100% ██████████ 282/282 [01:11<00:00, 3.94it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.11it/		
s]		all	1066	1066	0.716	0.112	0.0891	0.0574
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size		
6/100	4.53G Class	1.422 Images	2.242 Instances	1.52 Box(P)	9 R	640: 100% ██████████ 282/282 [01:11<00:00, 3.94it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.17it/		
s]		all	1066	1066	0.733	0.114	0.124	0.0804
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size		
7/100	4.53G Class	1.376 Images	2.219 Instances	1.488 Box(P)	8 R	640: 100% ██████████ 282/282 [01:11<00:00, 3.93it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.25it/		
s]		all	1066	1066	0.624	0.126	0.107	0.07
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size		
8/100	4.53G Class	1.338 Images	2.132 Instances	1.459 Box(P)	5 R	640: 100% ██████████ 282/282 [01:11<00:00, 3.92it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.14it/		
s]		all	1066	1066	0.69	0.196	0.156	0.0965
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size		
9/100	4.53G Class	1.293 Images	2.081 Instances	1.428 Box(P)	11 R	640: 100% ██████████ 282/282 [01:11<00:00, 3.92it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.19it/		
s]		all	1066	1066	0.395	0.187	0.14	0.092
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size		
10/100	4.53G Class	1.265 Images	2.039 Instances	1.412 Box(P)	8 R	640: 100% ██████████ 282/282 [01:11<00:00, 3.92it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.13it/		
s]		all	1066	1066	0.461	0.248	0.174	0.119

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
11/100	4.53G Class	1.247 Images	2.026 Instances	1.392 Box(P)	12 R	640: 100% ██████████ 282/282 [01:11<00:00, 3.92it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.21it/s]
12/100	4.53G Class	1.219 Images	1.952 Instances	1.373 Box(P)	12 R	640: 100% ██████████ 282/282 [01:11<00:00, 3.92it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.08it/s]
13/100	4.53G Class	1.219 Images	1.929 Instances	1.364 Box(P)	7 R	640: 100% ██████████ 282/282 [01:12<00:00, 3.91it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.25it/s]
14/100	4.53G Class	1.169 Images	1.855 Instances	1.33 Box(P)	6 R	640: 100% ██████████ 282/282 [01:11<00:00, 3.93it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.13it/s]
15/100	4.53G Class	1.175 Images	1.862 Instances	1.333 Box(P)	8 R	640: 100% ██████████ 282/282 [01:11<00:00, 3.93it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.25it/s]
16/100	4.53G Class	1.145 Images	1.827 Instances	1.311 Box(P)	5 R	640: 100% ██████████ 282/282 [01:11<00:00, 3.92it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.16it/s]
17/100	4.53G Class	1.132 Images	1.801 Instances	1.305 Box(P)	10 R	640: 100% ██████████ 282/282 [01:12<00:00, 3.91it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.25it/s]
all 1066 1066 0.33 0.35 0.243 0.169							
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	

18/100	4.53G Class	1.124 Images	1.775 Instances	1.294 Box(P)	7 R	640: 100% ██████████ 282/282 [01:12<00:00, 3.92it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.15it/s]
		all	1066	1066	0.421	0.282
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
19/100	4.53G Class	1.113 Images	1.762 Instances	1.285 Box(P)	9 R	640: 100% ██████████ 282/282 [01:12<00:00, 3.91it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.27it/s]
		all	1066	1066	0.288	0.337
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
20/100	4.53G Class	1.089 Images	1.743 Instances	1.27 Box(P)	9 R	640: 100% ██████████ 282/282 [01:12<00:00, 3.91it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.23it/s]
		all	1066	1066	0.49	0.339
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
21/100	4.53G Class	1.07 Images	1.725 Instances	1.256 Box(P)	11 R	640: 100% ██████████ 282/282 [01:12<00:00, 3.92it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.29it/s]
		all	1066	1066	0.575	0.243
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
22/100	4.53G Class	1.085 Images	1.695 Instances	1.268 Box(P)	12 R	640: 100% ██████████ 282/282 [01:11<00:00, 3.93it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.24it/s]
		all	1066	1066	0.443	0.278
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
23/100	4.53G Class	1.041 Images	1.673 Instances	1.244 Box(P)	4 R	640: 100% ██████████ 282/282 [01:11<00:00, 3.92it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.26it/s]
		all	1066	1066	0.321	0.381
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
24/100	4.53G Class	1.058 Images	1.684 Instances	1.248 Box(P)	10 R	640: 100% ██████████ 282/282 [01:12<00:00, 3.92it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.29it/s]
		all	1066	1066	0.506	0.339
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size

25/100	4.53G Class	1.043 Images	1.65 Instances	1.239 Box(P)	9 R	640: 100% ██████████ 282/282 [01:12<00:00, 3.92it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.33it/s]
		all	1066	1066	0.243	0.376
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
26/100	4.53G Class	1.004 Images	1.594 Instances	1.215 Box(P)	9 R	640: 100% ██████████ 282/282 [01:11<00:00, 3.93it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.26it/s]
		all	1066	1066	0.338	0.375
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
27/100	4.53G Class	1.024 Images	1.646 Instances	1.223 Box(P)	7 R	640: 100% ██████████ 282/282 [01:11<00:00, 3.92it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.35it/s]
		all	1066	1066	0.386	0.281
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
28/100	4.53G Class	0.9985 Images	1.607 Instances	1.21 Box(P)	9 R	640: 100% ██████████ 282/282 [01:11<00:00, 3.92it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.33it/s]
		all	1066	1066	0.466	0.349
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
29/100	4.53G Class	0.9956 Images	1.588 Instances	1.206 Box(P)	11 R	640: 100% ██████████ 282/282 [01:11<00:00, 3.93it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.29it/s]
		all	1066	1066	0.425	0.374
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
30/100	4.53G Class	0.994 Images	1.59 Instances	1.206 Box(P)	9 R	640: 100% ██████████ 282/282 [01:11<00:00, 3.93it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.38it/s]
		all	1066	1066	0.341	0.389
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
31/100	4.53G Class	0.9869 Images	1.575 Instances	1.195 Box(P)	8 R	640: 100% ██████████ 282/282 [01:12<00:00, 3.92it/s] mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.31it/s]
		all	1066	1066	0.419	0.41
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size

32/100	4.53G	0.9588	1.529	1.179	9	640: 100% ██████████ 282/282 [01:11<00:00, 3.93it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.30it/s]
s]	Class	Images	Instances	Box(P)	R		
	all	1066	1066	0.242	0.39	0.28	0.216
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
33/100	4.53G	0.9705	1.541	1.185	5	640: 100% ██████████ 282/282 [01:11<00:00, 3.92it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.22it/s]
s]	Class	Images	Instances	Box(P)	R		
	all	1066	1066	0.396	0.389	0.322	0.245
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
34/100	4.53G	0.958	1.517	1.183	6	640: 100% ██████████ 282/282 [01:11<00:00, 3.92it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.18it/s]
s]	Class	Images	Instances	Box(P)	R		
	all	1066	1066	0.347	0.383	0.311	0.244
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
35/100	4.53G	0.9581	1.541	1.184	10	640: 100% ██████████ 282/282 [01:11<00:00, 3.92it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.18it/s]
s]	Class	Images	Instances	Box(P)	R		
	all	1066	1066	0.331	0.41	0.29	0.221
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
36/100	4.53G	0.9421	1.508	1.171	5	640: 100% ██████████ 282/282 [01:12<00:00, 3.90it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.27it/s]
s]	Class	Images	Instances	Box(P)	R		
	all	1066	1066	0.26	0.417	0.3	0.234
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
37/100	4.53G	0.9267	1.483	1.158	7	640: 100% ██████████ 282/282 [01:12<00:00, 3.91it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.14it/s]
s]	Class	Images	Instances	Box(P)	R		
	all	1066	1066	0.478	0.356	0.308	0.24
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
38/100	4.53G	0.9275	1.468	1.157	10	640: 100% ██████████ 282/282 [01:12<00:00, 3.91it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.40it/s]
s]	Class	Images	Instances	Box(P)	R		
	all	1066	1066	0.468	0.346	0.321	0.252
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	

39/100	4.53G	0.929	1.482	1.156	7	640: 100% ██████████ 282/282 [01:12<00:00, 3.91it/s]	
s]	Class	Images	Instances	Box(P)	R	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.21it/	
	all	1066	1066	0.434	0.39	0.326	0.241
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
40/100	4.53G	0.9107	1.433	1.147	5	640: 100% ██████████ 282/282 [01:12<00:00, 3.91it/s]	
s]	Class	Images	Instances	Box(P)	R	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.20it/	
	all	1066	1066	0.352	0.418	0.311	0.238
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
41/100	4.53G	0.9006	1.426	1.141	9	640: 100% ██████████ 282/282 [01:11<00:00, 3.92it/s]	
s]	Class	Images	Instances	Box(P)	R	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.13it/	
	all	1066	1066	0.285	0.382	0.321	0.254
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
42/100	4.53G	0.9028	1.436	1.14	8	640: 100% ██████████ 282/282 [01:11<00:00, 3.93it/s]	
s]	Class	Images	Instances	Box(P)	R	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.30it/	
	all	1066	1066	0.372	0.356	0.334	0.262
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
43/100	4.53G	0.8907	1.418	1.135	7	640: 100% ██████████ 282/282 [01:11<00:00, 3.92it/s]	
s]	Class	Images	Instances	Box(P)	R	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.23it/	
	all	1066	1066	0.411	0.459	0.341	0.265
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
44/100	4.53G	0.8859	1.416	1.128	5	640: 100% ██████████ 282/282 [01:12<00:00, 3.92it/s]	
s]	Class	Images	Instances	Box(P)	R	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.38it/	
	all	1066	1066	0.347	0.452	0.354	0.275
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
45/100	4.53G	0.8805	1.381	1.13	7	640: 100% ██████████ 282/282 [01:11<00:00, 3.92it/s]	
s]	Class	Images	Instances	Box(P)	R	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.29it/	
	all	1066	1066	0.288	0.447	0.331	0.257
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	

46/100	4.53G	0.8838	1.387	1.126	6	640: 100% ██████████ 282/282 [01:12<00:00, 3.91it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.33it/s]
Class	Images	Instances	Box(P)	R			
	all	1066	1066	0.448	0.343	0.305	0.236
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
47/100	4.53G	0.8759	1.373	1.119	5	640: 100% ██████████ 282/282 [01:11<00:00, 3.92it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.39it/s]
Class	Images	Instances	Box(P)	R			
	all	1066	1066	0.393	0.357	0.329	0.257
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
48/100	4.53G	0.8776	1.393	1.124	6	640: 100% ██████████ 282/282 [01:12<00:00, 3.92it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.29it/s]
Class	Images	Instances	Box(P)	R			
	all	1066	1066	0.274	0.456	0.323	0.254
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
49/100	4.53G	0.8612	1.352	1.115	11	640: 100% ██████████ 282/282 [01:11<00:00, 3.92it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.28it/s]
Class	Images	Instances	Box(P)	R			
	all	1066	1066	0.403	0.451	0.36	0.281
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
50/100	4.53G	0.8492	1.337	1.105	8	640: 100% ██████████ 282/282 [01:12<00:00, 3.91it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.25it/s]
Class	Images	Instances	Box(P)	R			
	all	1066	1066	0.325	0.422	0.353	0.271
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
51/100	4.53G	0.8587	1.326	1.116	9	640: 100% ██████████ 282/282 [01:12<00:00, 3.91it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.31it/s]
Class	Images	Instances	Box(P)	R			
	all	1066	1066	0.452	0.379	0.353	0.283
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
52/100	4.53G	0.8389	1.334	1.099	9	640: 100% ██████████ 282/282 [01:12<00:00, 3.91it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.28it/s]
Class	Images	Instances	Box(P)	R			
	all	1066	1066	0.435	0.39	0.339	0.269
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	

53/100	4.53G	0.8477	1.309	1.107	4	640: 100% ██████████ 282/282 [01:12<00:00, 3.91it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.32it/s]
s]	Class	Images	Instances	Box(P)	R		
	all	1066	1066	0.365	0.411	0.354	0.276
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
54/100	4.53G	0.8392	1.309	1.107	4	640: 100% ██████████ 282/282 [01:11<00:00, 3.92it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.32it/s]
s]	Class	Images	Instances	Box(P)	R		
	all	1066	1066	0.364	0.4	0.354	0.278
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
55/100	4.53G	0.8323	1.305	1.095	5	640: 100% ██████████ 282/282 [01:12<00:00, 3.91it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.29it/s]
s]	Class	Images	Instances	Box(P)	R		
	all	1066	1066	0.276	0.427	0.327	0.253
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
56/100	4.53G	0.8401	1.293	1.095	6	640: 100% ██████████ 282/282 [01:12<00:00, 3.90it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.43it/s]
s]	Class	Images	Instances	Box(P)	R		
	all	1066	1066	0.469	0.346	0.337	0.269
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
57/100	4.53G	0.8336	1.29	1.089	6	640: 100% ██████████ 282/282 [01:11<00:00, 3.93it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.35it/s]
s]	Class	Images	Instances	Box(P)	R		
	all	1066	1066	0.288	0.417	0.348	0.275
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
58/100	4.53G	0.8166	1.266	1.087	11	640: 100% ██████████ 282/282 [01:11<00:00, 3.92it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.19it/s]
s]	Class	Images	Instances	Box(P)	R		
	all	1066	1066	0.341	0.431	0.349	0.271
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
59/100	4.53G	0.8017	1.245	1.073	9	640: 100% ██████████ 282/282 [01:12<00:00, 3.91it/s]	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.29it/s]
s]	Class	Images	Instances	Box(P)	R		
	all	1066	1066	0.477	0.39	0.353	0.268
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	

60/100	4.53G	0.808	1.234	1.079	7	640: 100% ██████████ 282/282 [01:11<00:00, 3.92it/s]
s]	Class	Images	Instances	Box(P)	R	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.20it/
	all	1066	1066	0.307	0.404	0.336 0.264
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
61/100	4.53G	0.7981	1.24	1.076	4	640: 100% ██████████ 282/282 [01:12<00:00, 3.92it/s]
s]	Class	Images	Instances	Box(P)	R	mAP50 mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 5.34it/
	all	1066	1066	0.498	0.362	0.347 0.268

EarlyStopping: Training stopped early as no improvement observed in last 10 epochs. Best results observed at epoch 51, best model saved as best.pt.

To update EarlyStopping(patience=10) pass a new patience value, i.e. `patience=300` or use `patience=0` to disable EarlyStopping.

61 epochs completed in 1.337 hours.

Optimizer stripped from runs/detect/train3/weights/last.pt, 18.5MB

Optimizer stripped from runs/detect/train3/weights/best.pt, 18.5MB

Validating runs/detect/train3/weights/best.pt...

Ultralytics 8.3.107 🚀 Python-3.11.11 torch-2.5.1+cu124 CUDA:0 (Tesla T4, 15095MiB)

YOLOv5s summary (fused): 84 layers, 9,115,793 parameters, 0 gradients, 23.8 GFLOPs

Class	Images	Instances	Box(P)	R	mAP50	mAP50-95): 100% ██████████ 34/34 [00:06<00:00, 4.95it/
s]	all	1066	1066	0.452	0.378	0.353 0.283
pickup_truck	154	154	0.436	0.558	0.488	0.424
bus	51	51	0.573	0.784	0.63	0.519
motorized_vehicle	74	74	0.181	0.0811	0.0714	0.0482
car	667	667	0.444	0.594	0.518	0.428
articulated_truck	28	28	0.367	0.56	0.465	0.37
work_van	39	39	0.491	0.308	0.359	0.305
pedestrian	13	13	0.293	0.154	0.136	0.0641
non-motorized_vehicle	6	6	1	0	0.00137	0.000683
bicycle	8	8	0.418	0.272	0.382	0.294
single_unit_truck	18	18	0.356	0.222	0.173	0.134
motorcycle	8	8	0.419	0.625	0.663	0.527

Speed: 0.1ms preprocess, 2.7ms inference, 0.0ms loss, 0.9ms postprocess per image

Results saved to runs/detect/train3

```
from glob import glob
def get_res_dir():
    res_dir_count = len(glob('runs/detect/*'))
```

```

res_dir = f'runs/detect/train{res_dir_count}'
return res_dir

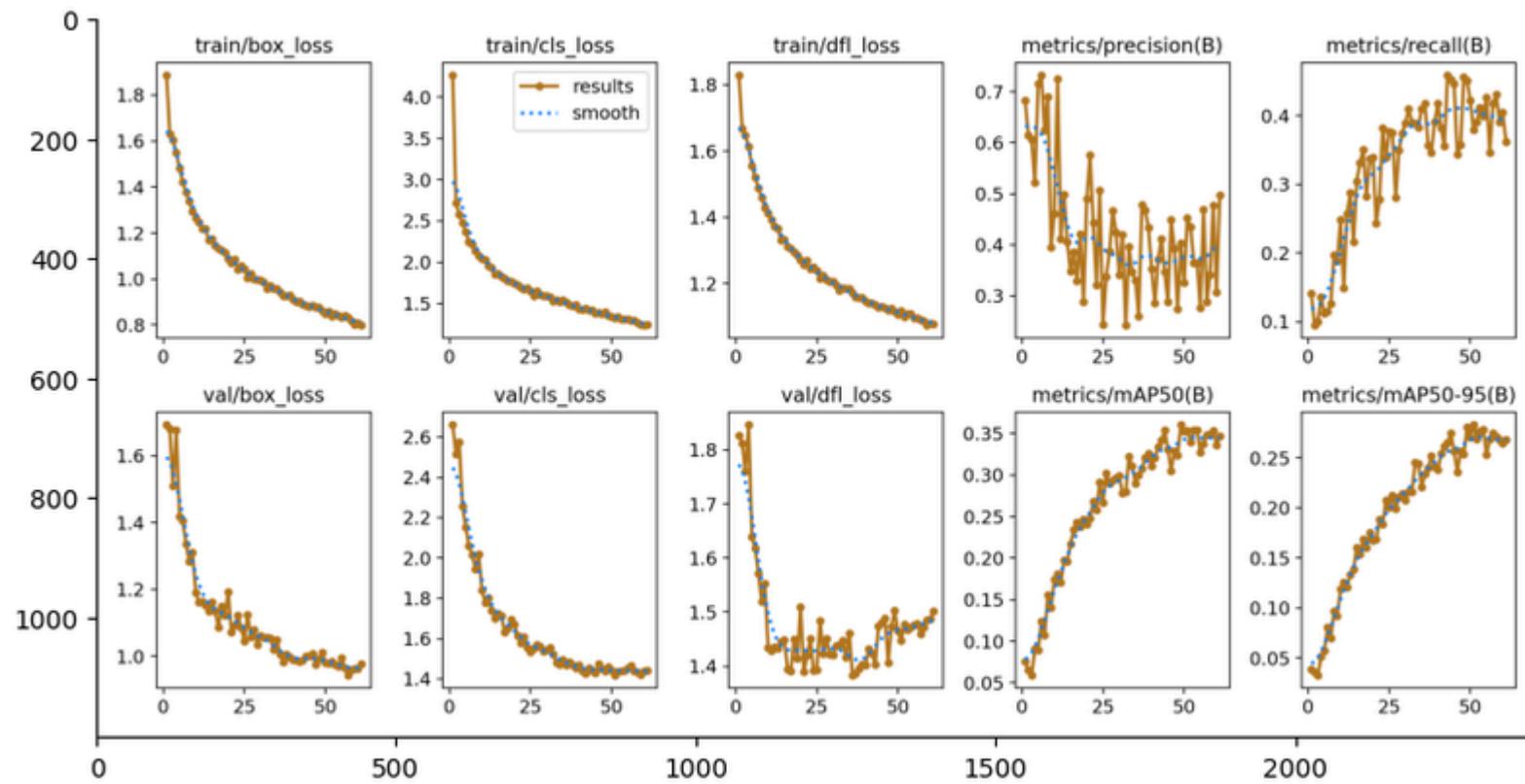
```

Precision, Recall and Accuracy curves

```

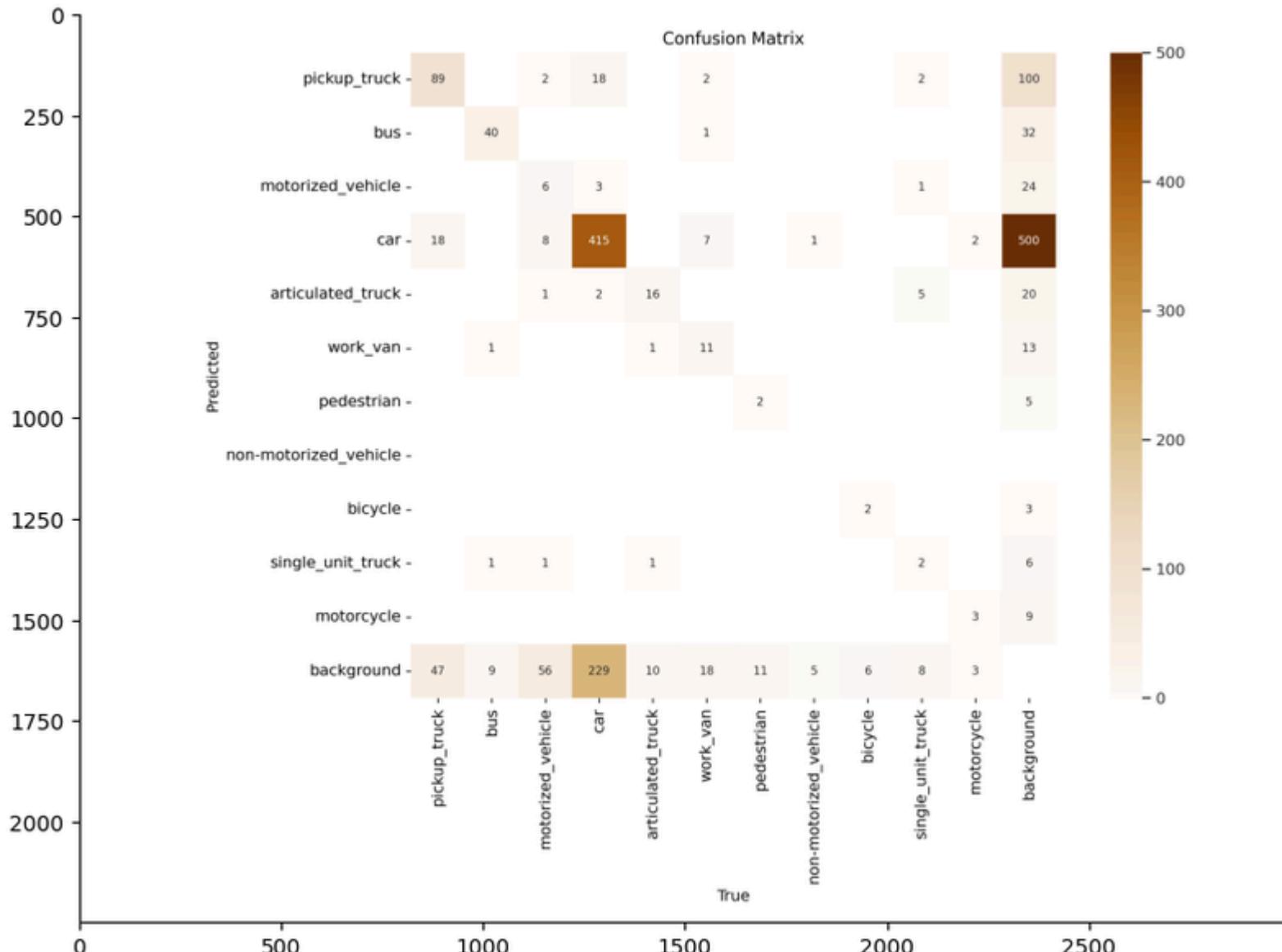
plt.figure(figsize = (10,10))
res_dir = get_res_dir()
img = cv2.imread(f'{res_dir}/results.png')
plt.gca().spines[['top', 'right']].set_visible(False)
_=plt.imshow(img)

```



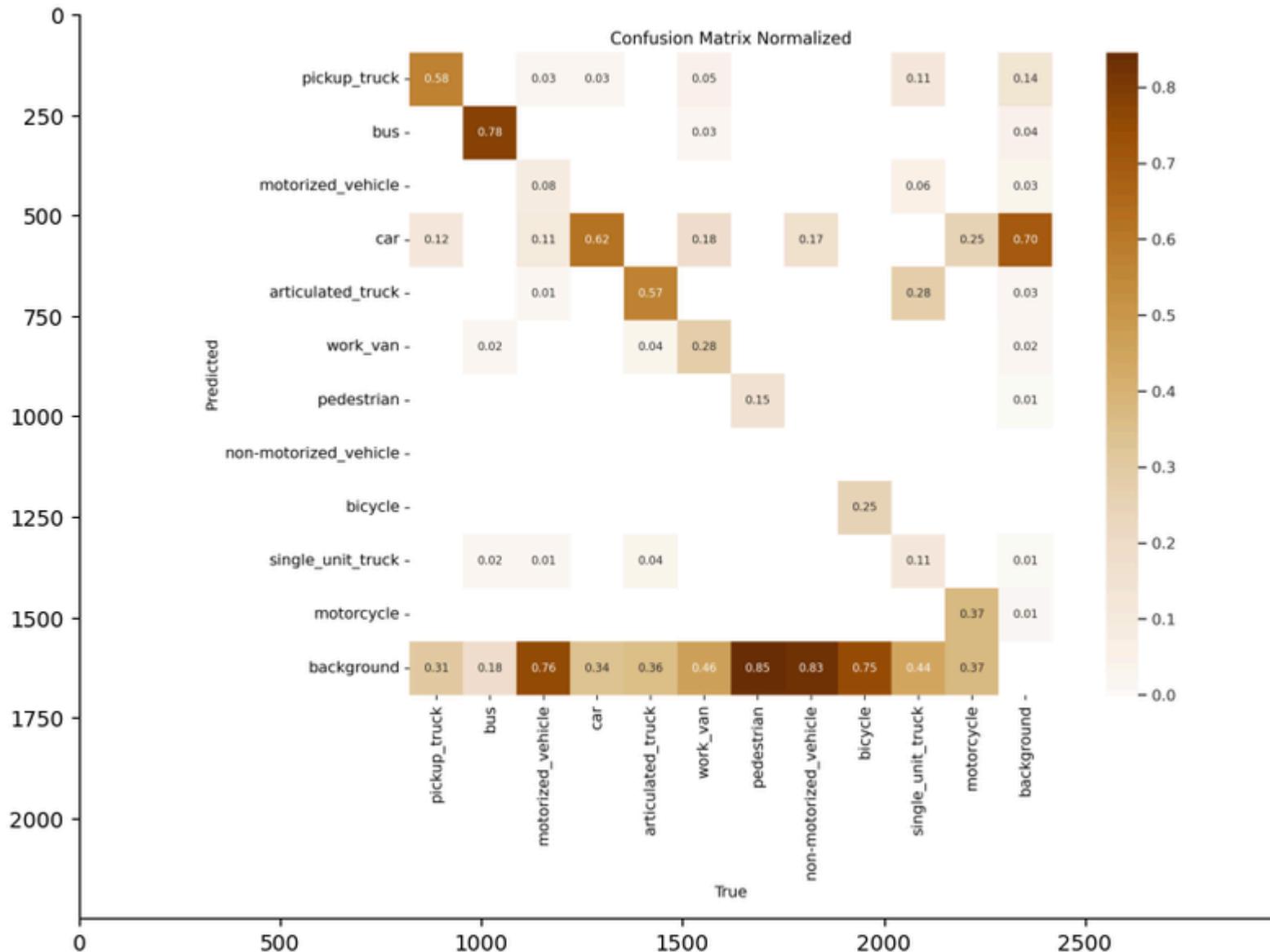
Confusion Matrix

```
plt.figure(figsize = (10,10))
img = cv2.imread(f'{res_dir}/confusion_matrix.png')
plt.gca().spines[['top', 'right',]].set_visible(False)
_=plt.imshow(img)
```



Confusion Matrix Normalized

```
plt.figure(figsize = (10,10))
img = cv2.imread(f'{res_dir}/confusion_matrix_normalized.png')
plt.gca().spines[['top', 'right',]].set_visible(False)
_=plt.imshow(img)
```



Derive predictions

```
results = model.predict(test_image_path)
```

image 1/60 /kaggle/working/datasets/test/images/00005598.jpg: 448x640 1 work_van, 39.6ms
image 2/60 /kaggle/working/datasets/test/images/00005599.jpg: 448x640 1 bus, 11.7ms
image 3/60 /kaggle/working/datasets/test/images/00005600.jpg: 448x640 1 car, 11.7ms
image 4/60 /kaggle/working/datasets/test/images/00005601.jpg: 448x640 (no detections), 11.7ms
image 5/60 /kaggle/working/datasets/test/images/00005602.jpg: 448x640 1 car, 11.7ms
image 6/60 /kaggle/working/datasets/test/images/00005603.jpg: 448x640 (no detections), 11.7ms
image 7/60 /kaggle/working/datasets/test/images/00005604.jpg: 448x640 1 pickup_truck, 1 car, 11.7ms
image 8/60 /kaggle/working/datasets/test/images/00005605.jpg: 448x640 1 pickup_truck, 11.8ms
image 9/60 /kaggle/working/datasets/test/images/00005606.jpg: 448x640 1 car, 11.7ms
image 10/60 /kaggle/working/datasets/test/images/00005607.jpg: 448x640 2 cars, 11.7ms
image 11/60 /kaggle/working/datasets/test/images/00005608.jpg: 448x640 (no detections), 11.7ms
image 12/60 /kaggle/working/datasets/test/images/00005609.jpg: 448x640 1 bus, 10.1ms
image 13/60 /kaggle/working/datasets/test/images/00005610.jpg: 448x640 1 car, 8.3ms
image 14/60 /kaggle/working/datasets/test/images/00005611.jpg: 448x640 1 pickup_truck, 3 cars, 1 work_van, 8.3ms
image 15/60 /kaggle/working/datasets/test/images/00005612.jpg: 448x640 2 pickup_trucks, 8.3ms
image 16/60 /kaggle/working/datasets/test/images/00005613.jpg: 448x640 2 cars, 8.3ms
image 17/60 /kaggle/working/datasets/test/images/00005614.jpg: 448x640 (no detections), 8.3ms
image 18/60 /kaggle/working/datasets/test/images/00005615.jpg: 448x640 1 motorized_vehicle, 2 cars, 8.4ms
image 19/60 /kaggle/working/datasets/test/images/00005616.jpg: 448x640 1 car, 8.3ms
image 20/60 /kaggle/working/datasets/test/images/00005617.jpg: 448x640 1 pickup_truck, 1 car, 8.3ms
image 21/60 /kaggle/working/datasets/test/images/00005618.jpg: 448x640 (no detections), 8.3ms
image 22/60 /kaggle/working/datasets/test/images/00005619.jpg: 448x640 1 car, 8.3ms
image 23/60 /kaggle/working/datasets/test/images/00005620.jpg: 448x640 1 car, 8.3ms
image 24/60 /kaggle/working/datasets/test/images/00005621.jpg: 448x640 (no detections), 8.3ms
image 25/60 /kaggle/working/datasets/test/images/00005622.jpg: 448x640 2 cars, 8.3ms
image 26/60 /kaggle/working/datasets/test/images/00005623.jpg: 448x640 1 car, 8.3ms
image 27/60 /kaggle/working/datasets/test/images/00005624.jpg: 448x640 2 cars, 8.0ms
image 28/60 /kaggle/working/datasets/test/images/00005625.jpg: 448x640 1 car, 6.4ms
image 29/60 /kaggle/working/datasets/test/images/00005626.jpg: 448x640 1 bus, 6.7ms
image 30/60 /kaggle/working/datasets/test/images/00005627.jpg: 448x640 (no detections), 6.4ms
image 31/60 /kaggle/working/datasets/test/images/00005628.jpg: 448x640 2 cars, 1 pedestrian, 6.4ms
image 32/60 /kaggle/working/datasets/test/images/00005629.jpg: 448x640 3 cars, 6.6ms
image 33/60 /kaggle/working/datasets/test/images/00005630.jpg: 448x640 1 pickup_truck, 6.6ms
image 34/60 /kaggle/working/datasets/test/images/00005631.jpg: 448x640 1 pickup_truck, 1 car, 6.5ms
image 35/60 /kaggle/working/datasets/test/images/00005632.jpg: 448x640 (no detections), 6.5ms
image 36/60 /kaggle/working/datasets/test/images/00005633.jpg: 448x640 1 car, 7.9ms
image 37/60 /kaggle/working/datasets/test/images/00005634.jpg: 448x640 1 car, 6.5ms
image 38/60 /kaggle/working/datasets/test/images/00005635.jpg: 448x640 1 car, 6.5ms
image 39/60 /kaggle/working/datasets/test/images/00005636.jpg: 448x640 1 bus, 2 cars, 6.7ms
image 40/60 /kaggle/working/datasets/test/images/00005637.jpg: 448x640 (no detections), 6.7ms
image 41/60 /kaggle/working/datasets/test/images/00005638.jpg: 448x640 2 articulated_trucks, 6.4ms

```
image 42/60 /kaggle/working/datasets/test/images/00005639.jpg: 448x640 (no detections), 6.5ms
image 43/60 /kaggle/working/datasets/test/images/00005640.jpg: 448x640 4 cars, 6.5ms
image 44/60 /kaggle/working/datasets/test/images/00005641.jpg: 448x640 1 bus, 6.4ms
image 45/60 /kaggle/working/datasets/test/images/00005642.jpg: 448x640 2 cars, 6.4ms
image 46/60 /kaggle/working/datasets/test/images/00005643.jpg: 448x640 2 cars, 6.4ms
image 47/60 /kaggle/working/datasets/test/images/00005644.jpg: 448x640 2 cars, 6.4ms
image 48/60 /kaggle/working/datasets/test/images/00005645.jpg: 448x640 1 car, 6.5ms
image 49/60 /kaggle/working/datasets/test/images/00005646.jpg: 448x640 1 pickup_truck, 2 cars, 6.4ms
image 50/60 /kaggle/working/datasets/test/images/00005647.jpg: 448x640 1 pickup_truck, 6.6ms
image 51/60 /kaggle/working/datasets/test/images/00005648.jpg: 448x640 1 car, 6.6ms
image 52/60 /kaggle/working/datasets/test/images/00005649.jpg: 448x640 1 pickup_truck, 1 bus, 6.2ms
image 53/60 /kaggle/working/datasets/test/images/00005650.jpg: 448x640 1 car, 6.2ms
image 54/60 /kaggle/working/datasets/test/images/00005651.jpg: 448x640 (no detections), 6.3ms
image 55/60 /kaggle/working/datasets/test/images/00005652.jpg: 448x640 (no detections), 6.6ms
image 56/60 /kaggle/working/datasets/test/images/00005653.jpg: 448x640 2 cars, 6.9ms
image 57/60 /kaggle/working/datasets/test/images/00005654.jpg: 448x640 (no detections), 7.0ms
image 58/60 /kaggle/working/datasets/test/images/00005655.jpg: 448x640 1 car, 6.2ms
image 59/60 /kaggle/working/datasets/test/images/00005656.jpg: 448x640 2 cars, 6.2ms
image 60/60 /kaggle/working/datasets/test/images/00005657.jpg: 448x640 2 cars, 6.2ms
Speed: 1.7ms preprocess, 8.5ms inference, 0.9ms postprocess per image at shape (1, 3, 448, 640)
```

```
def plot_box(image, boxes, labels):
    h,w,_ = image.shape
    for box_num,box in enumerate(boxes):
        xyxy = box.xyxy.int().tolist()
        xmin,ymin,xmax,ymax = xyxy[0][0], xyxy[0][1],xyxy[0][2],xyxy[0][3]
        width = xmax-xmin
        height = ymax-ymin
        class_name = class_names[int(labels[box_num])]
        color = COLORS[int(labels[box_num])]
        cv2.rectangle(image,(xmin,ymin),(xmax,ymax),color,5)
        font_scale = 1#min(1,max(3,int(w/500)))
        font_thickness = 2#min(2,max(10,int(w/50)))
        font = cv2.FONT_HERSHEY_SIMPLEX
        # p1,p1 = (int(xmin+1.5*width),ymin), (int(xmin+width),ymin-int(1.5*height))
        text_width = cv2.getTextSize(class_name,font,font_scale,font_thickness)[0][0]
        text_height = cv2.getTextSize(class_name,font,font_scale,font_thickness)[0][1]
        cv2.putText(image,class_name,(xmin,ymin+20),font,font_scale,color,font_thickness)

    return image
```

```

def get_identified_names(image, boxes, labels):
    h,w,_ = image.shape
    detections = []
    for box_num,box in enumerate(boxes):
        xyxy = box.xyxy.int().tolist()
        xmin,ymin,xmax,ymax = xyxy[0][0], xyxy[0][1],xyxy[0][2],xyxy[0][3]
        width = xmax-xmin
        height = ymax-ymin
        class_name = class_names[int(labels[box_num])]
        detections.append(class_name)

    return detections

```

Detections table

```

detects = []
paths = []
num_images = []

for res in results:
    image = res.orig_img
    boxes = res.boxes
    confidences = boxes.conf
    classes = boxes.cls
    path = res.path
    detections = get_identified_names(image,boxes,classes)
    detects.append(','.join(detections))
    paths.append(path)
    num_images.append(len(detections))
result_dict = {
    'detections' : detects,
    'paths': paths,
    'num_images': num_images
}
# pprint.pprint(result_dict)
result_df = pd.DataFrame(result_dict)
result_df

```

	detections	paths	num_images
0	work_van	/kaggle/working/datasets/test/images/00005598.jpg	1
1	bus	/kaggle/working/datasets/test/images/00005599.jpg	1
2	car	/kaggle/working/datasets/test/images/00005600.jpg	1
3		/kaggle/working/datasets/test/images/00005601.jpg	0
4	car	/kaggle/working/datasets/test/images/00005602.jpg	1
5		/kaggle/working/datasets/test/images/00005603.jpg	0
6	pickup_truck,car	/kaggle/working/datasets/test/images/00005604.jpg	2
7	pickup_truck	/kaggle/working/datasets/test/images/00005605.jpg	1
8	car	/kaggle/working/datasets/test/images/00005606.jpg	1
9	car,car	/kaggle/working/datasets/test/images/00005607.jpg	2
10		/kaggle/working/datasets/test/images/00005608.jpg	0
11	bus	/kaggle/working/datasets/test/images/00005609.jpg	1
12	car	/kaggle/working/datasets/test/images/00005610.jpg	1
13	car,car,work_van,pickup_truck,car	/kaggle/working/datasets/test/images/00005611.jpg	5
14	pickup_truck,pickup_truck	/kaggle/working/datasets/test/images/00005612.jpg	2
15	car,car	/kaggle/working/datasets/test/images/00005613.jpg	2
16		/kaggle/working/datasets/test/images/00005614.jpg	0
17	car,car,motorized_vehicle	/kaggle/working/datasets/test/images/00005615.jpg	3
18	car	/kaggle/working/datasets/test/images/00005616.jpg	1
19	pickup_truck,car	/kaggle/working/datasets/test/images/00005617.jpg	2
20		/kaggle/working/datasets/test/images/00005618.jpg	0
21	car	/kaggle/working/datasets/test/images/00005619.jpg	1

	detections		paths	num_images
22	car	/kaggle/working/datasets/test/images/00005620.jpg		1
23		/kaggle/working/datasets/test/images/00005621.jpg		0
24	car,car	/kaggle/working/datasets/test/images/00005622.jpg		2
25	car	/kaggle/working/datasets/test/images/00005623.jpg		1
26	car,car	/kaggle/working/datasets/test/images/00005624.jpg		2
27	car	/kaggle/working/datasets/test/images/00005625.jpg		1
28	bus	/kaggle/working/datasets/test/images/00005626.jpg		1
29		/kaggle/working/datasets/test/images/00005627.jpg		0
30	car,pedestrian,car	/kaggle/working/datasets/test/images/00005628.jpg		3
31	car,car,car	/kaggle/working/datasets/test/images/00005629.jpg		3
32	pickup_truck	/kaggle/working/datasets/test/images/00005630.jpg		1
33	car,pickup_truck	/kaggle/working/datasets/test/images/00005631.jpg		2
34		/kaggle/working/datasets/test/images/00005632.jpg		0
35	car	/kaggle/working/datasets/test/images/00005633.jpg		1
36	car	/kaggle/working/datasets/test/images/00005634.jpg		1
37	car	/kaggle/working/datasets/test/images/00005635.jpg		1
38	bus,car,car	/kaggle/working/datasets/test/images/00005636.jpg		3
39		/kaggle/working/datasets/test/images/00005637.jpg		0
40	articulated_truck,articulated_truck	/kaggle/working/datasets/test/images/00005638.jpg		2
41		/kaggle/working/datasets/test/images/00005639.jpg		0
42	car,car,car,car	/kaggle/working/datasets/test/images/00005640.jpg		4
43	bus	/kaggle/working/datasets/test/images/00005641.jpg		1

	detections	paths	num_images
44	car,car	/kaggle/working/datasets/test/images/00005642.jpg	2
45	car,car	/kaggle/working/datasets/test/images/00005643.jpg	2
46	car,car	/kaggle/working/datasets/test/images/00005644.jpg	2
47	car	/kaggle/working/datasets/test/images/00005645.jpg	1
48	car,pickup_truck,car	/kaggle/working/datasets/test/images/00005646.jpg	3
49	pickup_truck	/kaggle/working/datasets/test/images/00005647.jpg	1
50	car	/kaggle/working/datasets/test/images/00005648.jpg	1
51	bus,pickup_truck	/kaggle/working/datasets/test/images/00005649.jpg	2
52	car	/kaggle/working/datasets/test/images/00005650.jpg	1
53		/kaggle/working/datasets/test/images/00005651.jpg	0
54		/kaggle/working/datasets/test/images/00005652.jpg	0
55	car,car	/kaggle/working/datasets/test/images/00005653.jpg	2
56		/kaggle/working/datasets/test/images/00005654.jpg	0
57	car	/kaggle/working/datasets/test/images/00005655.jpg	1
58	car,car	/kaggle/working/datasets/test/images/00005656.jpg	2
59	car,car	/kaggle/working/datasets/test/images/00005657.jpg	2

Detections images

```

COLORS = np.random.uniform(0,255,size=(len(class_names),3))
slices = []
count = 0
d = list()
images_per_row = 5
for j,res in enumerate(results):

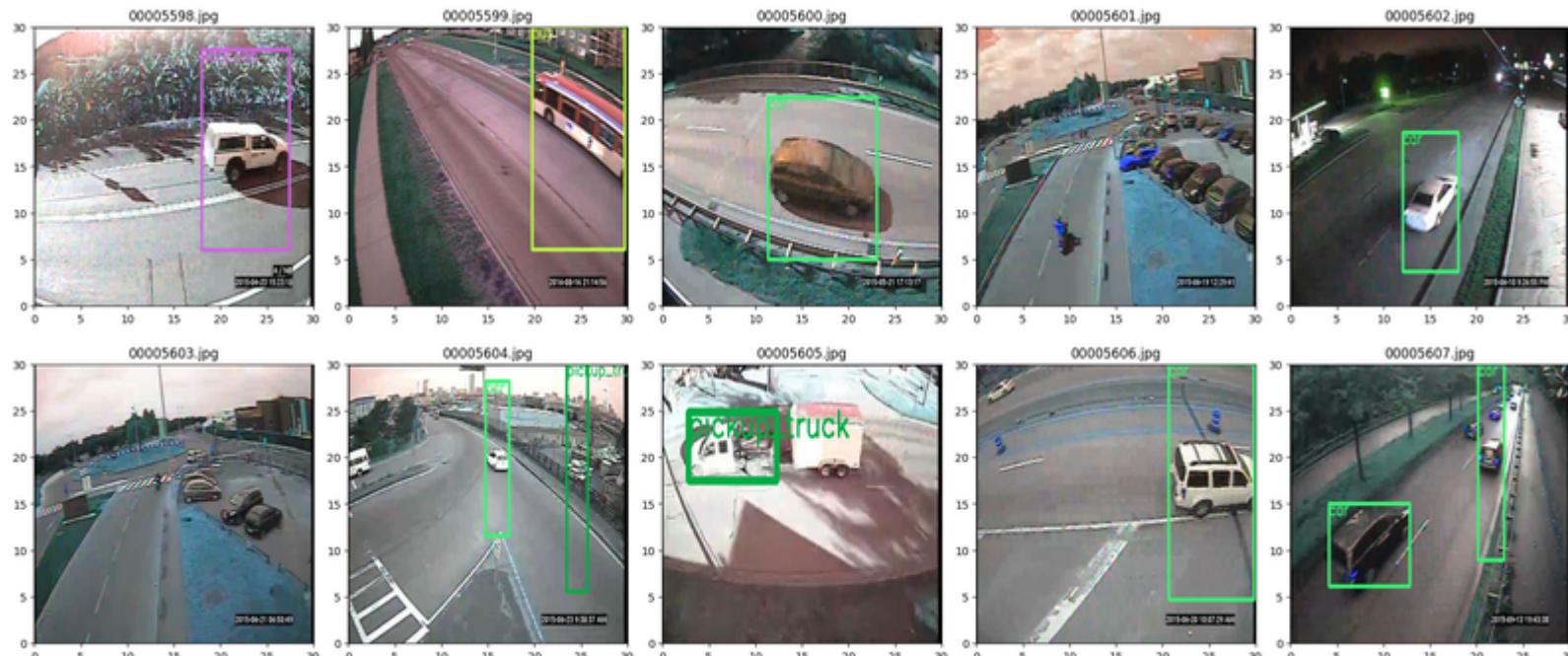
```

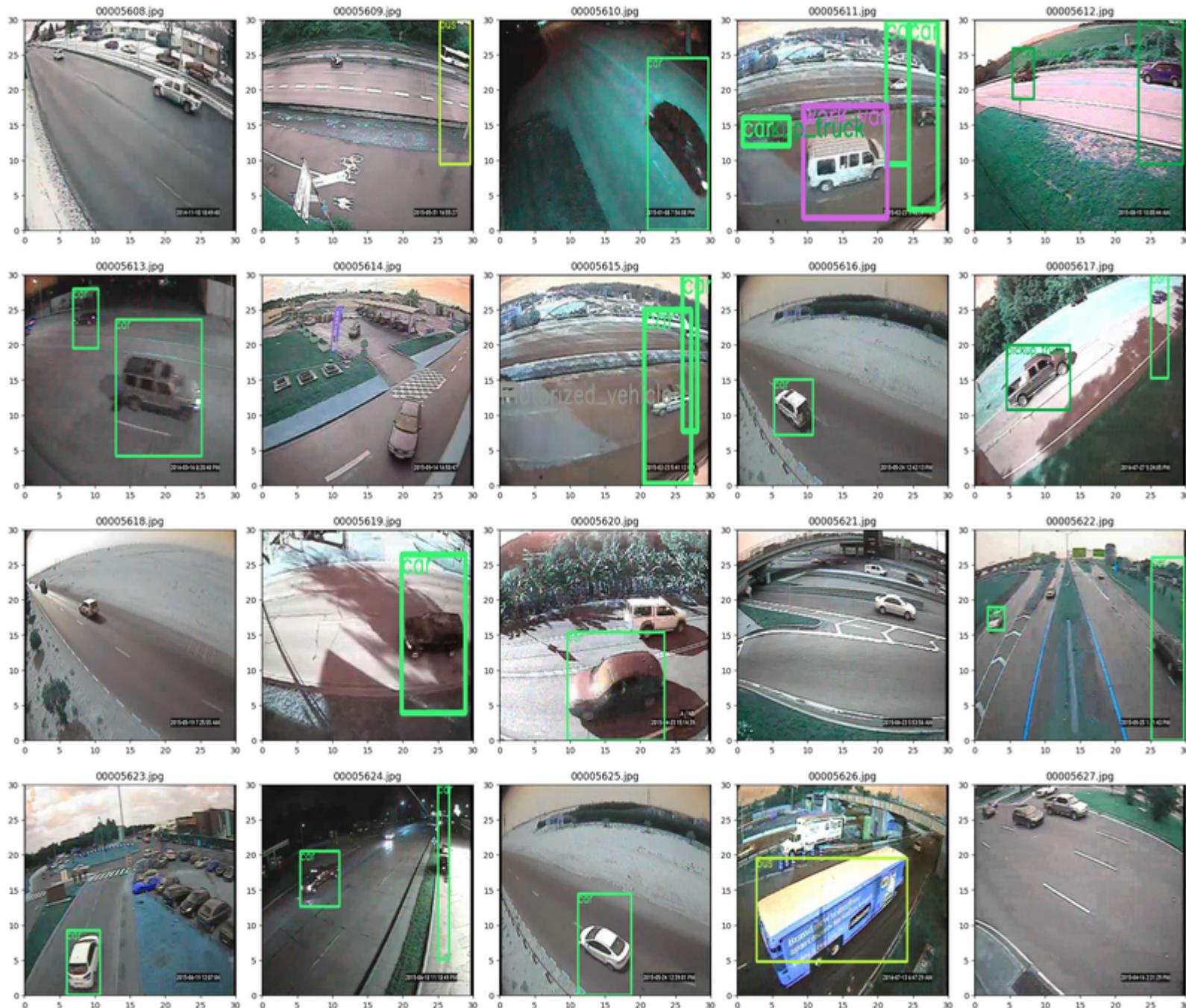
```

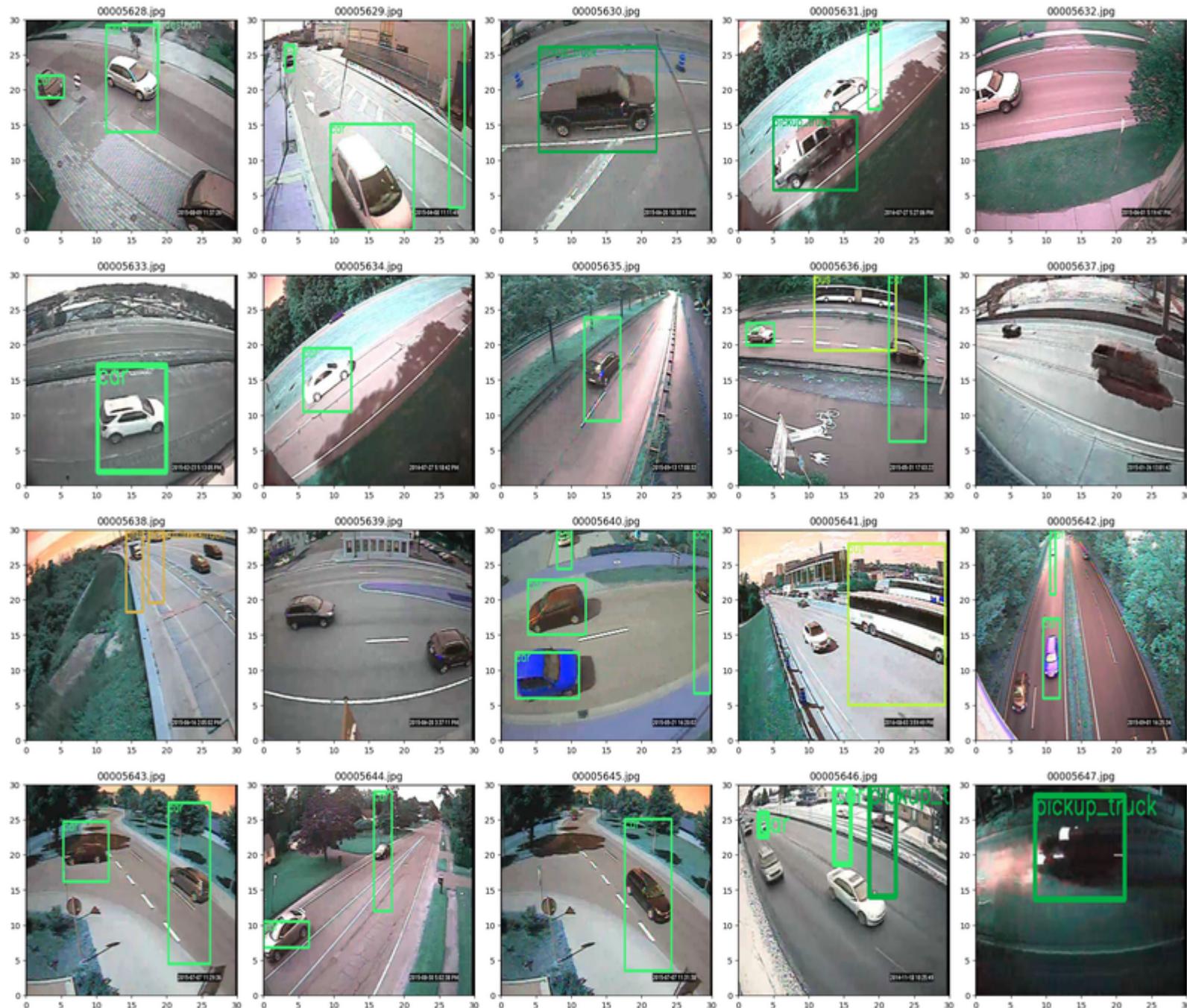
d.append(res)
if len(d) == images_per_row:
    slices.append(d)
    d = list()

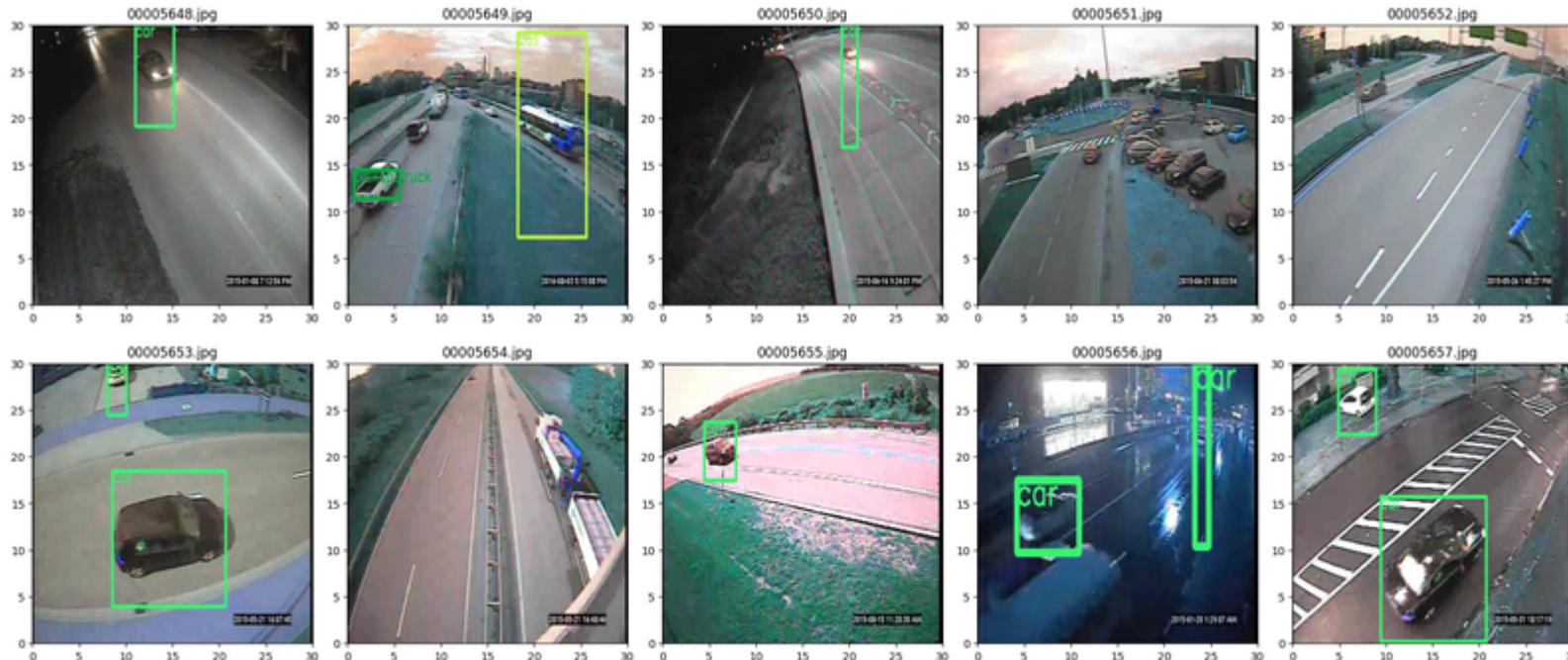
for i in (range(len(slices))):
    fig,axes = plt.subplots(nrows=1,ncols=images_per_row,figsize=(20,10),constrained_layout=True)
    results_ = slices[i]
    for j in range(len(results_)):
        res = results_[j]
        image = res.orig_img
        boxes = res.bboxes
        confidences = boxes.conf
        classes = boxes.cls
        path = res.path
        box_img = plot_box(image,boxes,classes)
        axes[j].imshow(box_img, extent=[0, 30, 0, 30])
        axes[j].set_title(path[-12:])

```









Part 2 : Tesla Auto pilot and Road Safety

Data wrangling

```

import pandas as pd
import torch
torch.device('cpu')
url = 'https://raw.githubusercontent.com/tksundar/autonomous_driving/refs/heads/master/Tesla%20-%20Deaths.csv'
data = pd.read_csv(url, skip_blank_lines=True, skipinitialspace=True)
data.isna().sum()
data.columns = data.columns.str.rstrip()
nan_counts = data.isna().sum()
columns_to_drop = ['Unnamed: 16', 'Unnamed: 17', 'Source', 'State']
columns_to_drop.extend(nan_counts[nan_counts >= 220].index)
data.drop(columns_to_drop, axis=1, inplace=True)
# Remove rows that does not have a case number

```

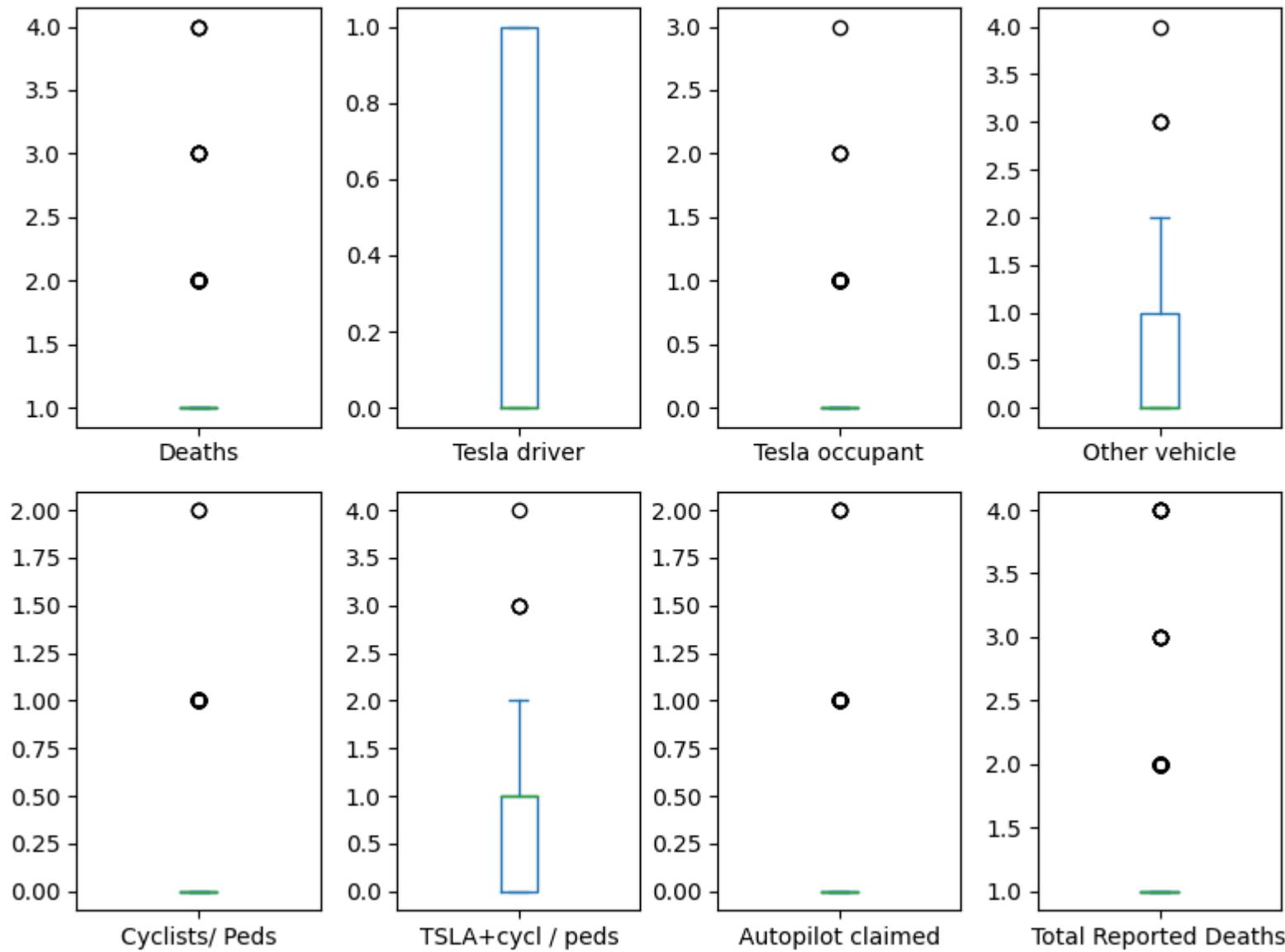
```
# The values in these rows are outliers
data = data.dropna(subset=['Case #'])
print(data.shape)
print(data.isna().sum())
```

```
(294, 15)
Case #                      0
Year                        0
Date                        0
Country                     0
Description                  0
Deaths                      0
Tesla driver                 5
Tesla occupant                9
Other vehicle                  4
Cyclists/ Peds                 3
TSLA+cycl / peds                2
Model                        0
Autopilot claimed                18
Verified Tesla Autopilot Deaths        4
Verified Tesla Autopilot Deaths + All Deaths Reported to NHTSA SGO    1
dtype: int64
```

```
import numpy as np
import matplotlib.pyplot as plt
columns = ['Deaths', 'Tesla driver', 'Tesla occupant', 'Other vehicle', 'Cyclists/ Peds',
           'TSLA+cycl / peds', 'Model', 'Autopilot claimed',
           'Verified Tesla Autopilot Deaths',
           'Verified Tesla Autopilot Deaths + All Deaths Reported to NHTSA SGO']
for col in columns:
    data[col] = data[col].apply(lambda x : '0' if x in ['-', ','] else x)
    data[col] = pd.to_numeric(data[col], errors='coerce')
    data[col] = data[col].fillna(0)
year_mode = data['Year'].mode()[0]
data['Year'] = data['Year'].apply(lambda x: year_mode if np.isnan(x) else x)
data['Year'] = data['Year'].apply(lambda x: year_mode if x == 202 else x)
data['Year'] = data['Year'].astype(int)
print(data.isna().sum())
```

```
Case #          0
Year           0
Date           0
Country        0
Description    0
Deaths         0
Tesla driver   0
Tesla occupant 0
Other vehicle  0
Cyclists/ Peds 0
TSLA+cycl / peds 0
Model          0
Autopilot claimed 0
Verified Tesla Autopilot Deaths 0
Verified Tesla Autopilot Deaths + All Deaths Reported to NHTSA SGO 0
dtype: int64
```

Distribution of numeric columns



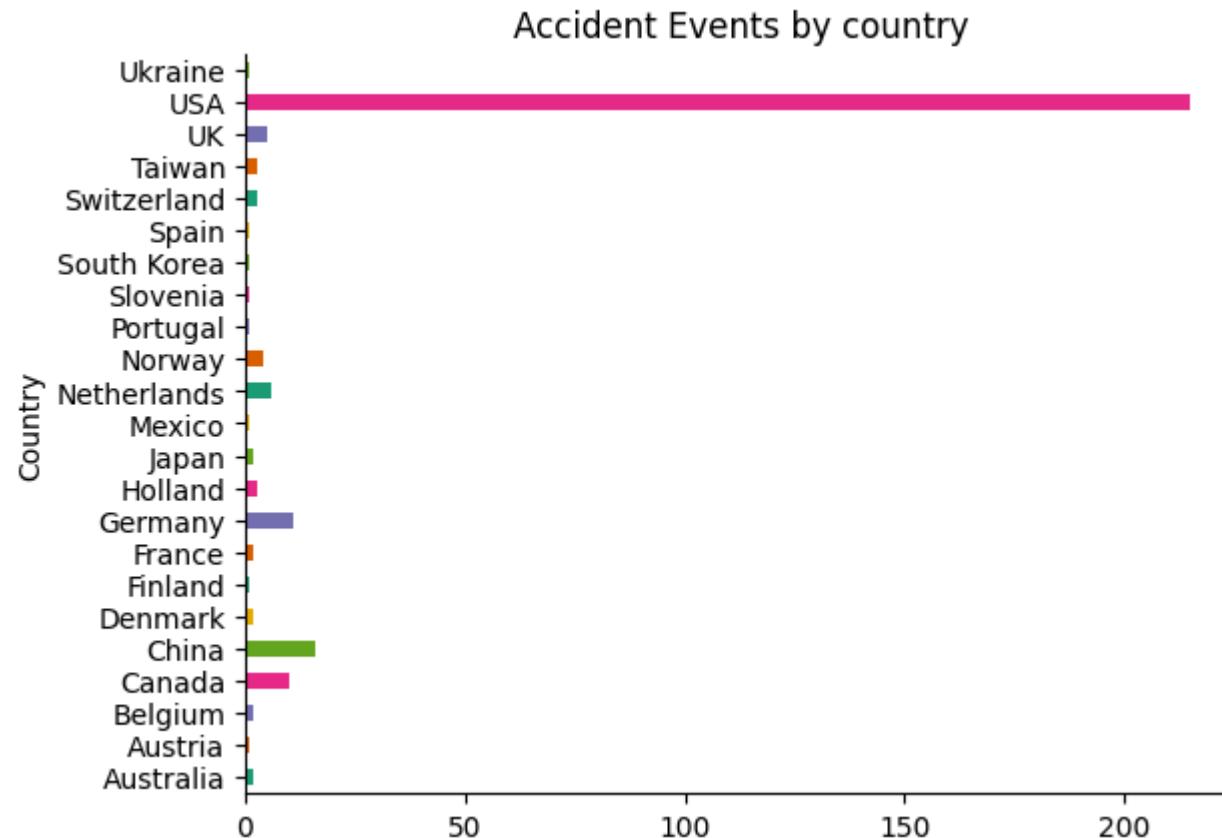
```
data.head()
```

Case #	Year	Date	Country	Description	Deaths	Tesla driver	Tesla occupant	Other vehicle	Cyclists/ Peds	TSLA+cycl / peds	Model	Autopilot claimed	Verified Tesla Autopilot Deaths	Deaths + All Deaths Reported to NHTSA SGO	To Report Death	
0	294.0	2022	2023-01-17	USA	Tesla crashes into back of semi	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
1	293.0	2022	2023-01-07	Canada	Tesla crashes	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
2	292.0	2022	2023-01-07	USA	Tesla hits pole, catches on fire	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
3	291.0	2022	2022-12-22	USA	Tesla crashes and burns	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
4	290.0	2022	2022-12-19	Canada	Tesla crashes into storefront	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0

Accident Events by country

```
from matplotlib import pyplot as plt
import seaborn as sns
data.groupby('Country').size().plot(kind='barh', color=sns.palettes.mpl_palette('Dark2'))
```

```
plt.gca().spines[['top', 'right']].set_visible(False)
_=plt.title('Accident Events by country')
```

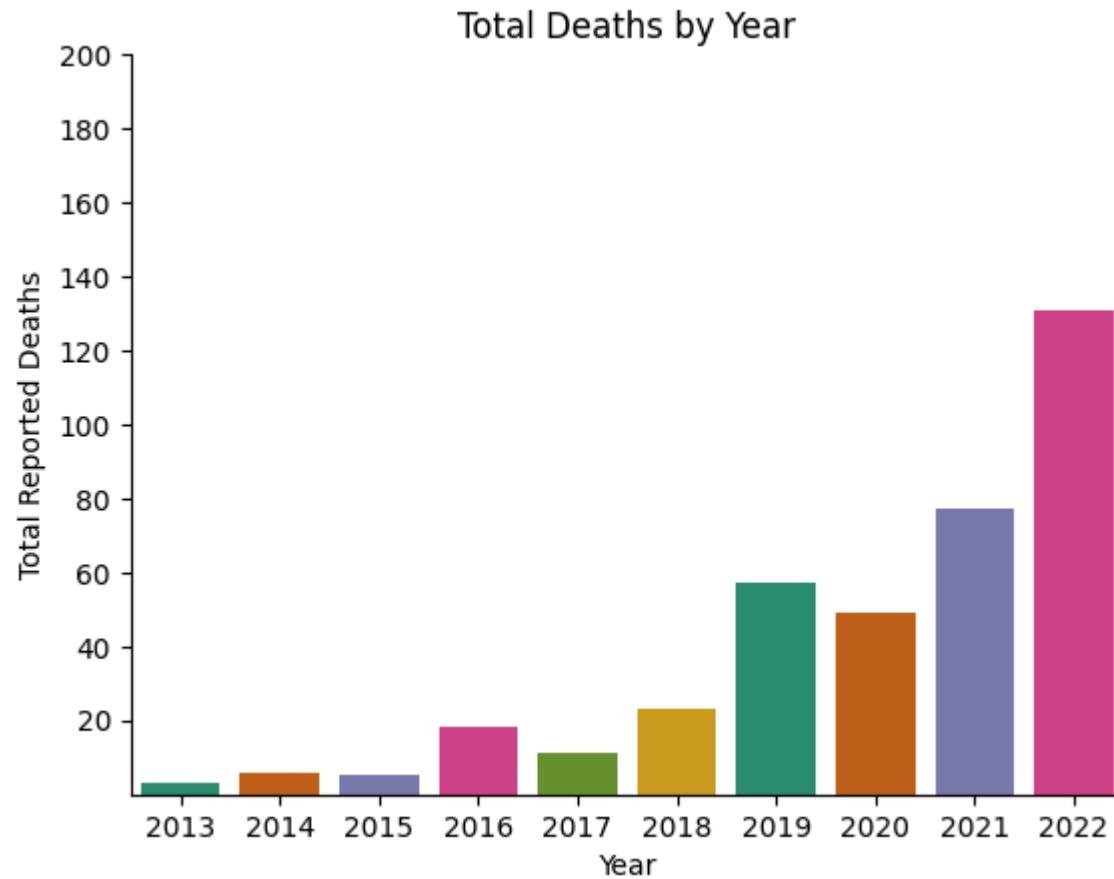


Deaths by year

```
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
deaths_by_year = pd.DataFrame(data.groupby('Year')['Total Reported Deaths'].sum()).reset_index()
plt.gca().spines[['top', 'right']].set_visible(False)

fig=sns.barplot(data = deaths_by_year, x='Year', y='Total Reported Deaths' , palette = sns.mpl_palette('Dark2'))
plt.title('Total Deaths by Year')
```

```
plt.yticks(np.arange(20,220,20))
plt.show()
```

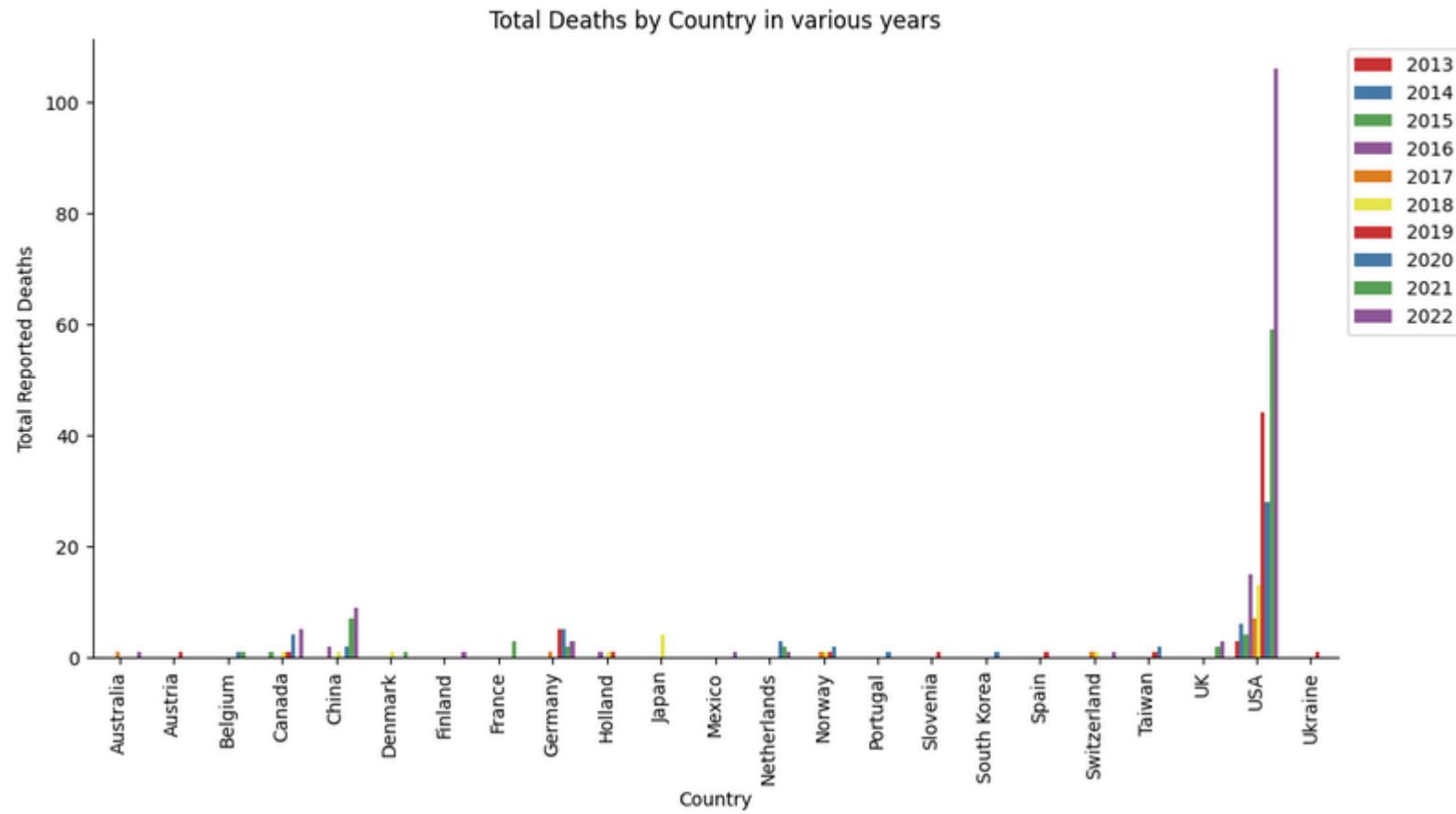


Deaths by country and year

```
deaths_by_year = pd.DataFrame(data.groupby(['Country','Year'])['Total Reported Deaths'].sum()).reset_index()

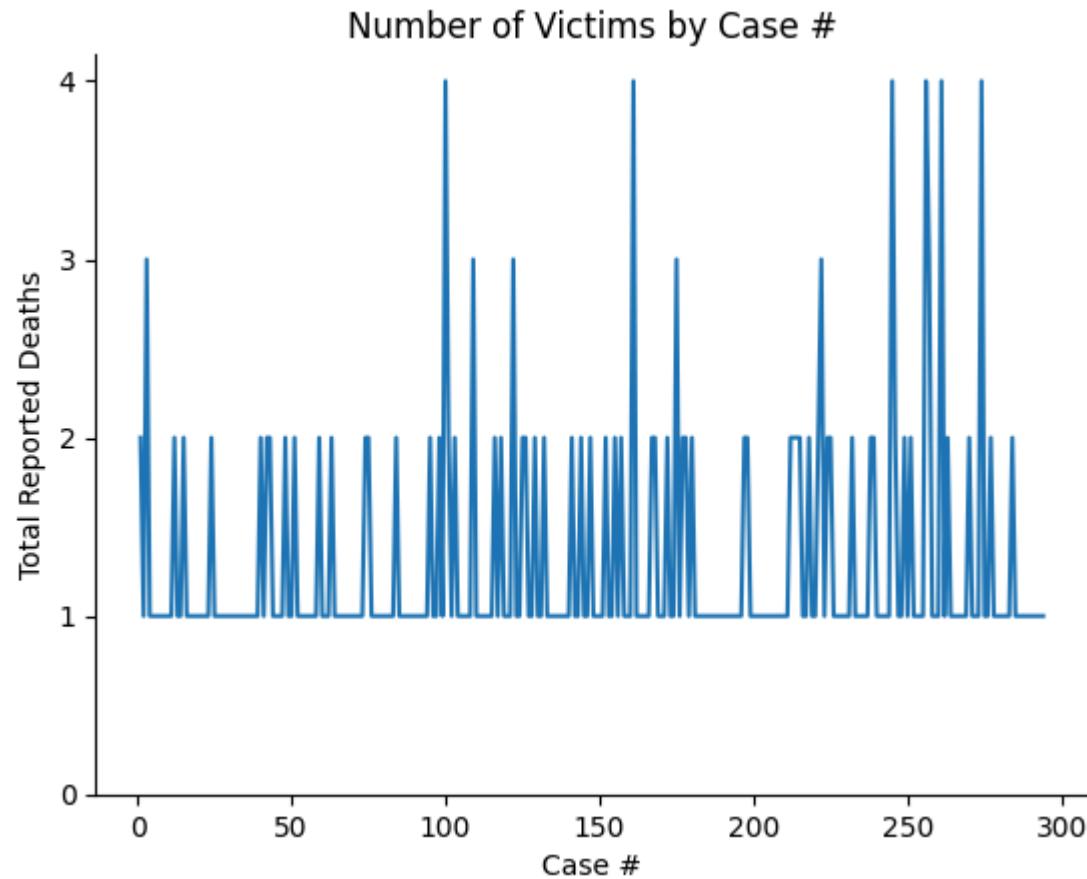
plt.figure(figsize = (12,6))
fig=sns.barplot(data = deaths_by_year, x='Country', hue = "Year", y='Total Reported Deaths', palette = sns.mpl_palette('Set1'))
plt.title('Total Deaths by Country in various years')
plt.xticks(rotation=90)
plt.legend(bbox_to_anchor=(1, 1), loc="upper left")
```

```
plt.gca().spines[['top', 'right']].set_visible(False)  
plt.show()
```



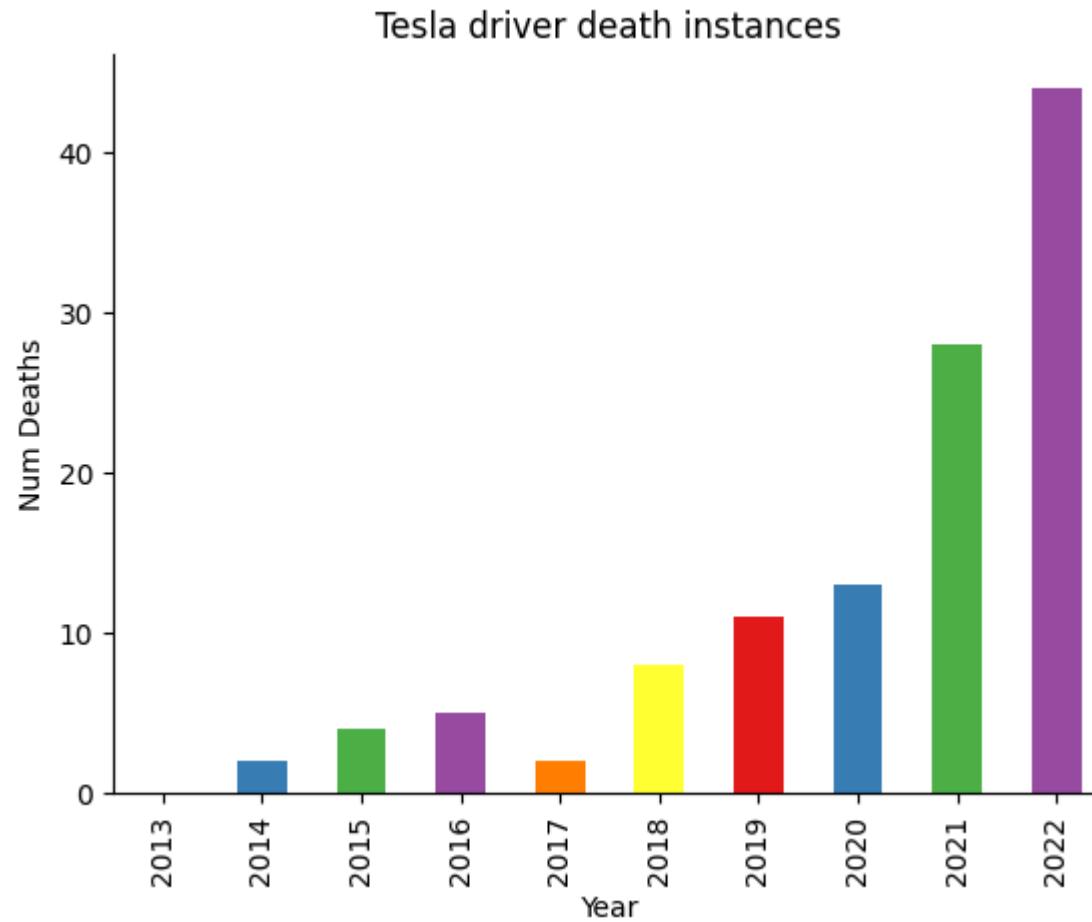
Number of victims per accident

```
df = data [['Case #', 'Total Reported Deaths']].sort_values(by='Case #')  
sns.lineplot(data=df , x= 'Case #' , y='Total Reported Deaths')  
plt.gca().spines[['top', 'right']].set_visible(False)  
plt.yticks(np.arange(0,5))  
_=plt.title('Number of Victims by Case #')
```



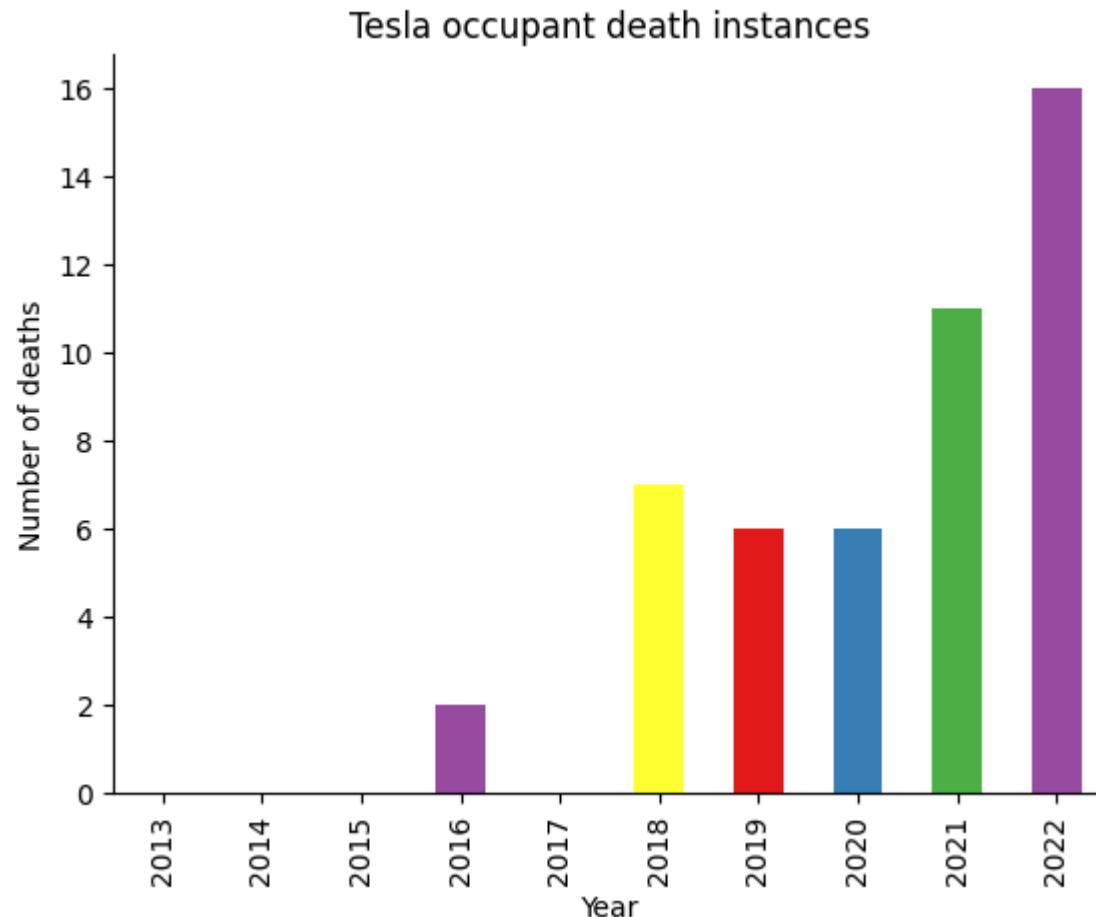
Instances of Tesla driver deaths

```
_=data.groupby('Year')['Tesla driver'].sum().plot(kind='bar', color = sns.mpl_palette('Set1'))
plt.title('Tesla driver death instances')
plt.xlabel('Year')
plt.ylabel('Num Deaths')
plt.gca().spines[['top', 'right']].set_visible(False)
```



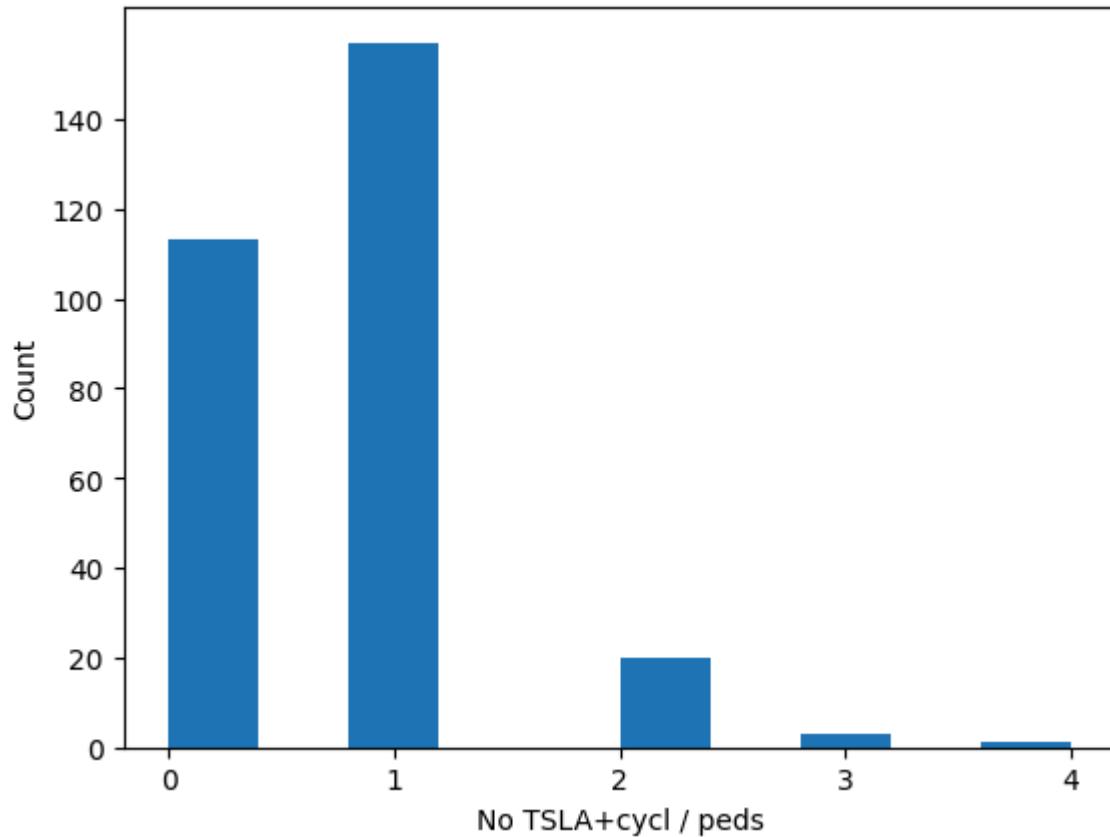
Instances of Tesla occupant deaths

```
data.groupby('Year')['Tesla occupant'].sum().plot(kind='bar', color = sns.mpl_palette('Set1'))
plt.title('Tesla occupant death instances')
plt.ylabel('Number of deaths')
plt.gca().spines[['top', 'right']].set_visible(False)
```



Distribution of events in which the vehicle hit a cyclist or a pedestrian

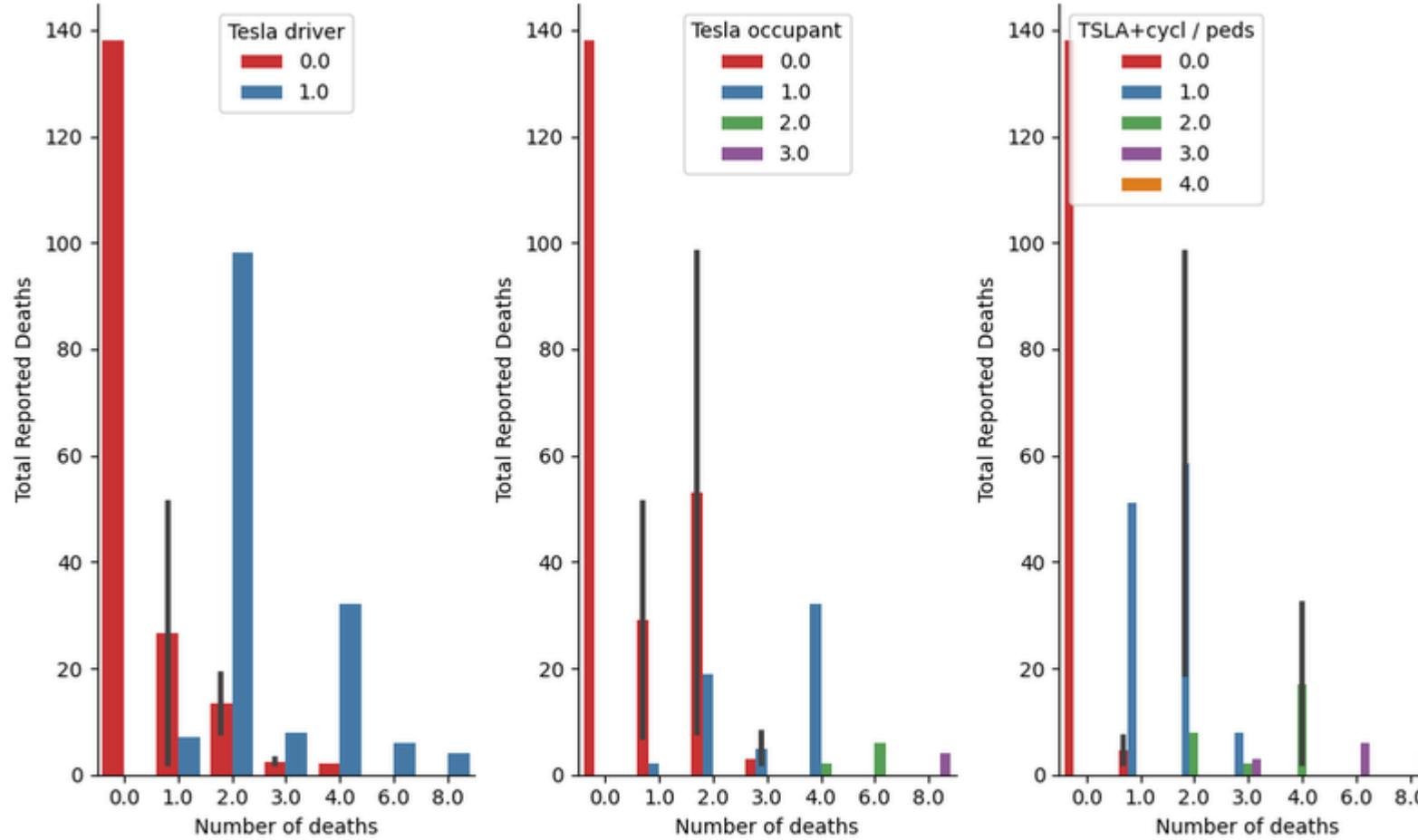
```
_=data['TSLA+cycl / peds'].hist(bins=10,grid=False)
_=plt.xlabel('No TSLA+cycl / peds')
_=plt.xticks(np.arange(0,5,1))
_=plt.ylabel('Count')
_=plt.yticks(np.arange(0,160,20))
```



Deaths of occupant, driver of a Tesla along with a cyclist or pedestrian

```
columns = ['Tesla driver', 'Tesla occupant', 'TSLA+cycl / peds']
grp = pd.DataFrame(data.groupby(columns)['Total Reported Deaths'].sum()).reset_index()
grp['Cumulative'] = grp.apply(lambda row: row['Tesla driver'] + row['Tesla occupant'] + row['TSLA+cycl / peds'], axis = 1)
plt.figure(figsize=(10,6))
for i,col in enumerate(columns):
    plt.subplot(1,3,i+1)
    sns.barplot(data = grp , x = 'Cumulative',y='Total Reported Deaths',hue=col,palette = sns.mpl_palette('Set1'))
    plt.xlabel('Number of deaths')
    plt.gca().spines[['top', 'right']].set_visible(False)
```

```
plt.tight_layout()
```



```
def check_for_two_vehicles(description):  
    vehicles = []  
    count = 0  
    description_lower = description.lower() # For case-insensitive matching  
    temp = []  
    for object_ in objects:  
        if object_.lower() in description_lower:  
            temp.append(object_)
```

```
if(len(temp)>= 2):
    return 1
return 0

url_labels = 'https://raw.githubusercontent.com/tksundar/autonomous_driving/refs/heads/master/labels.csv'
labels = pd.read_csv(url_labels, names = ['id','name','xmin','ymin','xmax','ymax'])

objects = labels['name'].unique().tolist()
objects.remove('pedestrian')
objects.append('Tesla')
objects.append('Semi')
```

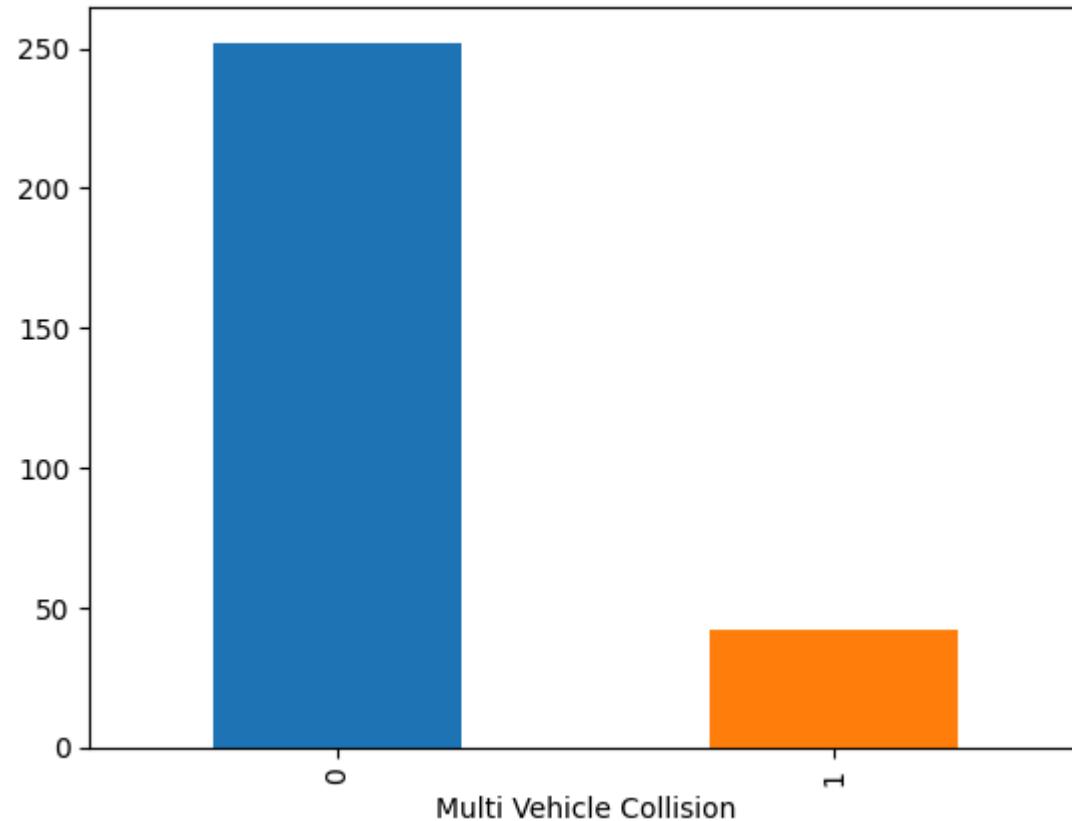
Frequency of Tesla colliding with other vehicles

```
data['Multi Vehicle Collision'] = data.apply(lambda row: check_for_two_vehicles(row['Description']),axis=1)
_=data.groupby(['Multi Vehicle Collision']).size().plot(kind = 'bar', color = sns.mpl_palette('tab10'))
plt.title('Frequency of multi vehicle collisions')

print(f"Total multi vehicle collisions : {data['Multi Vehicle Collision'].sum()}\n")
```

Total multi vehicle collisions : 42

Frequency of multi vehicle collisions



```
_=pd.crosstab(data['Other vehicle'],data['Multi Vehicle Collision']).plot(kind='bar')
plt.ylabel('Number of multi vehicle collisions')
_=pd.crosstab(data['Cyclists/ Peds'],data['Multi Vehicle Collision']).plot(kind='bar')
_=plt.ylabel('Number of multi vehicle collisions')
```

