In [1]:

```python
import numpy as np
import tensorflow as tf
from tensorflow import keras
from keras.applications import VGG16
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from keras.callbacks import ModelCheckpoint, EarlyStopping
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from ultralytics import YOLO
import matplotlib.pyplot as plt
import cv2
import pandas as pd
from pathlib import Path
import yaml
import shutil
import pickle

import os
```

```
2024-12-04 23:55:39.977758: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is op
timized to use available CPU instructions in performance-critical operations.
To enable the following instructions: SSE4.1 SSE4.2 AVX AVX2 FMA, in other operations, rebuild TensorFlow wit
h the appropriate compiler flags.
```

# Question 2

In [2]:

```python
df = pd.read_csv('./car_detection_dataset/train_bounding_boxes.csv')
image_files = sorted([l for l in os.listdir('./car_detection_dataset/training_images')
                      if l.lower().endswith(('.png', '.jpg', '.jpeg'))
                      and os.path.isfile(os.path.join('./car_detection_dataset/training_images', l))])

train_images, val_images = train_test_split(image_files, test_size=0.2, random_state=42)

os.makedirs('./car_detection_dataset/train/image', exist_ok=True)
os.makedirs('./car_detection_dataset/train/label', exist_ok=True)
os.makedirs('./car_detection_dataset/val/image', exist_ok=True)
os.makedirs('./car_detection_dataset/val/label', exist_ok=True)
```

In [3]:

```python
# Process and move training images
for images in train_images:
    source = os.path.join('./car_detection_dataset/training_images', images)

    # Copy image to the destination folder
    destination = os.path.join('./car_detection_dataset/train/images', images)
    shutil.copy(source, destination)

    # Retrieve bounding box data for the current image
    image_data = df[df['image'] == images]
    label_path = os.path.join('./car_detection_dataset/train/labels', f'{Path(images).stem}.txt')

    # Open the label file and write the bounding box annotations
    with open(label_path, 'w') as lp:
        if not image_data.empty:
            image = cv2.imread(source)
            for _, row in image_data.iterrows():
                # Normalize the bounding box coordinates
                x = (row['xmin'] + row['xmax']) / 2 / image.shape[1]
                y = (row['ymin'] + row['ymax']) / 2 / image.shape[0]
                w = (row['xmax'] - row['xmin']) / image.shape[1]
                h = (row['ymax'] - row['ymin']) / image.shape[0]

                # Write the annotation (label) in YOLO format
                lp.write(f'0 {x} {y} {w} {h}\n')
```

In [4]:

```python
# Process and move validation images
for images in val_images:
    source = os.path.join('./car_detection_dataset/training_images', images)
    # Copy image to the destination folder
    destination = os.path.join('./car_detection_dataset/val/images', images)
    shutil.copy(source, destination)

    # Retrieve bounding box data for the current image
    image_data = df[df['image'] == images]
    label_path = os.path.join('./car_detection_dataset/val/labels', f'{Path(images).stem}.txt')

    # Open the label file and write the bounding box annotations
    with open(label_path, 'w') as lp:
        if not image_data.empty:
            image = cv2.imread(source)
            for _, row in image_data.iterrows():
                # Normalize the bounding box coordinates
                x = (row['xmin'] + row['xmax']) / 2 / image.shape[1]
                y = (row['ymin'] + row['ymax']) / 2 / image.shape[0]
                w = (row['xmax'] - row['xmin']) / image.shape[1]
                h = (row['ymax'] - row['ymin']) / image.shape[0]

                # Write the annotation (label) in YOLO format
                lp.write(f'0 {x} {y} {w} {h}\n')
```

In [5]:

```python
data_yaml = {
    'train': './car_detection_dataset/train/images',
    'val': './car_detection_dataset/val/images',
    'nc': 1,  # Number of classes (in this case, only 'car')
    'names': ['car']
}

# Save the YAML configuration to a file
import yaml
with open('data.yaml', 'w') as f:
    yaml.dump(data_yaml, f)
```

In [6]:

```python
maxWidth, maxHeight = 0, 0
for name in os.listdir('./car_detection_dataset/training_images/'):
    imagePath = os.path.join('./car_detection_dataset/training_images/', name)
    # Read the image
    image = cv2.imread(imagePath)
    if image is not None:  # Ensure the image was read successfully
        height = image.shape[0]
        width =  image.shape[1] # Get the dimensions
        # Update max width and height if necessary
        if width > maxWidth:
            maxWidth = width
        if height > maxHeight:
            maxHeight = height

# Output the maximum width and height
print(f"Max Width: {maxWidth}, Max Height: {maxHeight}")
```

```
Max Width: 676, Max Height: 380
```

In [10]:

```python
# Define file paths
pickle_file = 'model_info.pkl'
dataset_yaml = 'data.yaml'
results_csv = ""
model_path = ""

# Set batch sizes for training
best_map = 0
best_model_info = {}

# Load dataset configuration from YAML file
with open(dataset_yaml, 'r') as f:
    dataset_config = yaml.safe_load(f)
```

```
epoch_options = [10, 20]
# Iterate over batch sizes to train and select the best model
for epoch in epoch_options:
    results_csv = f'./runs/detect/car_detection_{epoch}/results.csv'
    model_path = f'./runs/detect/car_detection_{epoch}/weights/best.pt'

    if not os.path.exists(model_path):  # If model hasn't been trained
        try:
            # Determine image size (based on dataset)
            max_width = max_height = 0
            image = max(32 * round(max(max_width, max_height) / 32), 640)

            # Initialize YOLOv5m model and start training
            model = YOLO('yolov5m.pt')
            model.train(
                data=dataset_yaml,
                epochs=epoch,
                imgsz=image,
                batch=16,
                name=f'car_detection_{epoch}'
            )
        except Exception as e:
            print(f"Error during training with batch size {epoch}: {e}")
            continue

    # Read the results CSV after training
    if os.path.exists(results_csv):
        df = pd.read_csv(results_csv)
        current_map = df['metrics/precision(B)'].iloc[-1]

        # Update best model if current mAP is higher
        if current_map > best_map:
            best_map = current_map
            best_model_info = {'model_path': model_path, 'results_csv': results_csv}
    else:
        print(f"Results file for batch {epoch} not found.")
```

```
PRO TIP 💡 Replace 'model=yolov5m.pt' with new 'model=yolov5mu.pt'.
YOLOv5 'u' models are trained with https://github.com/ultralytics/ultralytics and feature improved performanc
e vs standard YOLOv5 models trained with https://github.com/ultralytics/yolov5.

New https://pypi.org/project/ultralytics/8.3.41 available 😃 Update with 'pip install -U ultralytics'
Ultralytics 8.3.39 🚀 Python-3.12.7 torch-2.5.1+cu124 CUDA:0 (NVIDIA A100-SXM4-80GB, 81038MiB)
engine/trainer: task=detect, mode=train, model=yolov5m.pt, data=data.yaml, epochs=10, time=None, patience=10
0, batch=16, imgsz=640, save=True, save_period=-1, cache=False, device=None, workers=8, project=None, name=ca
r_detection_10, exist_ok=False, pretrained=True, optimizer=auto, verbose=True, seed=0, deterministic=True, si
ngle_cls=False, rect=False, cos_lr=False, close_mosaic=10, resume=False, amp=True, fraction=1.0, profile=Fals
e, freeze=None, multi_scale=False, overlap_mask=True, mask_ratio=4, dropout=0.0, val=True, split=val, save_js
on=False, save_hybrid=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_m
asks=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=Fal
se, int8=False, dynamic=False, simplify=True, opset=None, workspace=None, nms=False, lr0=0.01, lrf=0.01, mome
ntum=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, dfl=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, cop
y_paste_mode=flip, auto_augment=randaugment, erasing=0.4, crop_fraction=1.0, cfg=None, tracker=botsort.yaml,
save_dir=runs/detect/car_detection_10
Overriding model.yaml nc=80 with nc=1

                   from  n    params  module                                       arguments
  0                  -1  1      5280  ultralytics.nn.modules.conv.Conv             [3, 48, 6, 2, 2]
  1                  -1  1     41664  ultralytics.nn.modules.conv.Conv             [48, 96, 3, 2]
  2                  -1  2     65280  ultralytics.nn.modules.block.C3              [96, 96, 2]
  3                  -1  1    166272  ultralytics.nn.modules.conv.Conv             [96, 192, 3, 2]
  4                  -1  4    444672  ultralytics.nn.modules.block.C3              [192, 192, 4]
  5                  -1  1    664320  ultralytics.nn.modules.conv.Conv             [192, 384, 3, 2]
  6                  -1  6   2512896  ultralytics.nn.modules.block.C3              [384, 384, 6]
  7                  -1  1   2655744  ultralytics.nn.modules.conv.Conv             [384, 768, 3, 2]
  8                  -1  2   4134912  ultralytics.nn.modules.block.C3              [768, 768, 2]
  9                  -1  1   1476864  ultralytics.nn.modules.block.SPPF            [768, 768, 5]
 10                  -1  1    295680  ultralytics.nn.modules.conv.Conv             [768, 384, 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  2   1182720  ultralytics.nn.modules.block.C3              [768, 384, 2, False]
 14                  -1  1     74112  ultralytics.nn.modules.conv.Conv             [384, 192, 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  2    296448  ultralytics.nn.modules.block.C3              [384, 192, 2, False]
 18                  -1  1    332160  ultralytics.nn.modules.conv.Conv             [192, 192, 3, 2]
 19            [-1, 14]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 20                  -1  2   1035264  ultralytics.nn.modules.block.C3              [384, 384, 2, False]
 21                  -1  1   1327872  ultralytics.nn.modules.conv.Conv             [384, 384, 3, 2]
 22            [-1, 10]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 23                  -1  2   4134912  ultralytics.nn.modules.block.C3              [768, 768, 2, False]
 24        [17, 20, 23]  1   4218643  ultralytics.nn.modules.head.Detect           [1, [192, 384, 768]]
YOLOv5m summary: 339 layers, 25,065,715 parameters, 25,065,699 gradients, 64.4 GFLOPs

Transferred 553/559 items from pretrained weights
TensorBoard: Start with 'tensorboard --logdir runs/detect/car_detection_10', view at http://localhost:6006/
Freezing layer 'model.24.dfl.conv.weight'
AMP: running Automatic Mixed Precision (AMP) checks...
AMP: checks passed ✅
```

```
train: Scanning /blue/eel5934/r.anumula/project-3-graduate-RaviTejaAnumula/car_detection_dataset/train/label
s.cache... 961 images, 625 backgrounds, 0 corrupt: 100%|██████████| 961/961 [00:00<?, ?it/s]
/blue/eel5934/r.anumula/.conda/envs/art/lib/python3.12/site-packages/torch/utils/data/dataloader.py:617: User
Warning: This DataLoader will create 8 worker processes in total. Our suggested max number of worker in curre
nt system is 2, which is smaller than what this DataLoader is going to create. Please be aware that excessive
worker creation might get DataLoader running slow or even freeze, lower the worker number to avoid potential
slowness/freeze if necessary.
  warnings.warn(
val: Scanning /blue/eel5934/r.anumula/project-3-graduate-RaviTejaAnumula/car_detection_dataset/val/labels.cac
he... 362 images, 212 backgrounds, 0 corrupt: 100%|██████████| 362/362 [00:00<?, ?it/s]
/blue/eel5934/r.anumula/.conda/envs/art/lib/python3.12/site-packages/torch/utils/data/dataloader.py:617: User
Warning: This DataLoader will create 16 worker processes in total. Our suggested max number of worker in curr
ent system is 2, which is smaller than what this DataLoader is going to create. Please be aware that excessiv
e worker creation might get DataLoader running slow or even freeze, lower the worker number to avoid potentia
l slowness/freeze if necessary.
  warnings.warn(
Plotting labels to runs/detect/car_detection_10/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=0.002, momentum=0.9) with parameter groups 91 weight(decay=0.0), 98 weight(decay=0.0005),
97 bias(decay=0.0)
TensorBoard: model graph visualization added ✅
Image sizes 640 train, 640 val
Using 8 dataloader workers
Logging results to runs/detect/car_detection_10
Starting training for 10 epochs...
Closing dataloader mosaic
/blue/eel5934/r.anumula/.conda/envs/art/lib/python3.12/site-packages/torch/utils/data/dataloader.py:617: User
Warning: This DataLoader will create 8 worker processes in total. Our suggested max number of worker in curre
nt system is 2, which is smaller than what this DataLoader is going to create. Please be aware that excessive
worker creation might get DataLoader running slow or even freeze, lower the worker number to avoid potential
slowness/freeze if necessary.
  warnings.warn(
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       1/10      7.07G       1.55      3.891      1.466          0        640: 100%|██████████| 61/61 [00:27<
00:00,  2.23it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:08<00:00,  1.49it/s]
                   all        362        241      0.111       0.83      0.117     0.0699

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       2/10      6.99G      1.581      1.308      1.607          0        640: 100%|██████████| 61/61 [00:08<
00:00,  6.92it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00,  9.67it/s]
                   all        362        241      0.897      0.859      0.928      0.494

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       3/10      7.13G      1.593      1.145      1.694          1        640: 100%|██████████| 61/61 [00:08<
00:00,  7.03it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00, 10.48it/s]
                   all        362        241      0.894      0.838       0.92      0.525

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       4/10      7.03G      1.485      1.066      1.609          0        640: 100%|██████████| 61/61 [00:08<
00:00,  7.10it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00, 10.14it/s]
                   all        362        241      0.178      0.765      0.173     0.0991

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       5/10      7.13G      1.395     0.8885       1.52          1        640: 100%|██████████| 61/61 [00:08<
00:00,  7.03it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00, 10.24it/s]
                   all        362        241      0.888      0.888      0.945      0.562

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
```

```
       6/10      7.1G      1.433     0.8864      1.531          0        640: 100%|██████████| 61/61 [00:08<
00:00,  7.18it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00, 10.04it/s]
                   all        362        241      0.948      0.903      0.972      0.635

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       7/10     7.08G      1.373     0.7914      1.512          0        640: 100%|██████████| 61/61 [00:08<
00:00,  7.21it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00, 10.07it/s]
                   all        362        241      0.932      0.925      0.974      0.612

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       8/10        7G      1.305     0.7805      1.391          2        640: 100%|██████████| 61/61 [00:08<
00:00,  6.81it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00, 10.82it/s]
                   all        362        241      0.948      0.975      0.988      0.643

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       9/10     7.13G      1.293     0.7106      1.405          0        640: 100%|██████████| 61/61 [00:08<
00:00,  7.16it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00, 11.28it/s]
                   all        362        241       0.95      0.938      0.986      0.665

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      10/10     7.12G      1.248     0.6657      1.399          1        640: 100%|██████████| 61/61 [00:08<
00:00,  7.23it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00, 10.29it/s]
                   all        362        241      0.971      0.954      0.991      0.657

10 epochs completed in 0.039 hours.
Optimizer stripped from runs/detect/car_detection_10/weights/last.pt, 50.5MB
Optimizer stripped from runs/detect/car_detection_10/weights/best.pt, 50.5MB

Validating runs/detect/car_detection_10/weights/best.pt...
Ultralytics 8.3.39 🚀 Python-3.12.7 torch-2.5.1+cu124 CUDA:0 (NVIDIA A100-SXM4-80GB, 81038MiB)
YOLOv5m summary (fused): 248 layers, 25,045,795 parameters, 0 gradients, 64.0 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00,  8.33it/s]
```

```
                   all        362        241       0.95      0.938      0.986      0.664
Speed: 0.1ms preprocess, 1.0ms inference, 0.0ms loss, 0.6ms postprocess per image
Results saved to runs/detect/car_detection_10
PRO TIP 💡 Replace 'model=yolov5m.pt' with new 'model=yolov5mu.pt'.
YOLOv5 'u' models are trained with https://github.com/ultralytics/ultralytics and feature improved performance vs standard YOLOv5 models trained with https://github.com/ultralytics/yolov5.

New https://pypi.org/project/ultralytics/8.3.41 available 😃 Update with 'pip install -U ultralytics'
Ultralytics 8.3.39 🚀 Python-3.12.7 torch-2.5.1+cu124 CUDA:0 (NVIDIA A100-SXM4-80GB, 81038MiB)
engine/trainer: task=detect, mode=train, model=yolov5m.pt, data=data.yaml, epochs=20, time=None, patience=100, batch=16, imgsz=640, save=True, save_period=-1, cache=False, device=None, workers=8, project=None, name=car_detection_20, exist_ok=False, pretrained=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, save_hybrid=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, dfl=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/car_detection_20
Overriding model.yaml nc=80 with nc=1

                   from  n    params  module                                       arguments
  0                  -1  1      5280  ultralytics.nn.modules.conv.Conv             [3, 48, 6, 2, 2]
  1                  -1  1     41664  ultralytics.nn.modules.conv.Conv             [48, 96, 3, 2]
  2                  -1  2     65280  ultralytics.nn.modules.block.C3              [96, 96, 2]
  3                  -1  1    166272  ultralytics.nn.modules.conv.Conv             [96, 192, 3, 2]
  4                  -1  4    444672  ultralytics.nn.modules.block.C3              [192, 192, 4]
  5                  -1  1    664320  ultralytics.nn.modules.conv.Conv             [192, 384, 3, 2]
  6                  -1  6   2512896  ultralytics.nn.modules.block.C3              [384, 384, 6]
  7                  -1  1   2655744  ultralytics.nn.modules.conv.Conv             [384, 768, 3, 2]
  8                  -1  2   4134912  ultralytics.nn.modules.block.C3              [768, 768, 2]
  9                  -1  1   1476864  ultralytics.nn.modules.block.SPPF            [768, 768, 5]
 10                  -1  1    295680  ultralytics.nn.modules.conv.Conv             [768, 384, 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  2   1182720  ultralytics.nn.modules.block.C3              [768, 384, 2, False]
 14                  -1  1     74112  ultralytics.nn.modules.conv.Conv             [384, 192, 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  2    296448  ultralytics.nn.modules.block.C3              [384, 192, 2, False]
 18                  -1  1    332160  ultralytics.nn.modules.conv.Conv             [192, 192, 3, 2]
 19            [-1, 14]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 20                  -1  2   1035264  ultralytics.nn.modules.block.C3              [384, 384, 2, False]
 21                  -1  1   1327872  ultralytics.nn.modules.conv.Conv             [384, 384, 3, 2]
 22            [-1, 10]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 23                  -1  2   4134912  ultralytics.nn.modules.block.C3              [768, 768, 2, False]
 24        [17, 20, 23]  1   4218643  ultralytics.nn.modules.head.Detect           [1, [192, 384, 768]]
YOLOv5m summary: 339 layers, 25,065,715 parameters, 25,065,699 gradients, 64.4 GFLOPs

Transferred 553/559 items from pretrained weights
TensorBoard: Start with 'tensorboard --logdir runs/detect/car_detection_20', view at http://localhost:6006/
Freezing layer 'model.24.dfl.conv.weight'
AMP: running Automatic Mixed Precision (AMP) checks...
AMP: checks passed ✅
```

```
train: Scanning /blue/eel5934/r.anumula/project-3-graduate-RaviTejaAnumula/car_detection_dataset/train/label
s.cache... 961 images, 625 backgrounds, 0 corrupt: 100%|██████████| 961/961 [00:00<?, ?it/s]
/blue/eel5934/r.anumula/.conda/envs/art/lib/python3.12/site-packages/torch/utils/data/dataloader.py:617: User
Warning: This DataLoader will create 8 worker processes in total. Our suggested max number of worker in curre
nt system is 2, which is smaller than what this DataLoader is going to create. Please be aware that excessive
worker creation might get DataLoader running slow or even freeze, lower the worker number to avoid potential
slowness/freeze if necessary.
  warnings.warn(
val: Scanning /blue/eel5934/r.anumula/project-3-graduate-RaviTejaAnumula/car_detection_dataset/val/labels.cac
he... 362 images, 212 backgrounds, 0 corrupt: 100%|██████████| 362/362 [00:00<?, ?it/s]
/blue/eel5934/r.anumula/.conda/envs/art/lib/python3.12/site-packages/torch/utils/data/dataloader.py:617: User
Warning: This DataLoader will create 16 worker processes in total. Our suggested max number of worker in curr
ent system is 2, which is smaller than what this DataLoader is going to create. Please be aware that excessiv
e worker creation might get DataLoader running slow or even freeze, lower the worker number to avoid potentia
l slowness/freeze if necessary.
  warnings.warn(
Plotting labels to runs/detect/car_detection_20/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=0.002, momentum=0.9) with parameter groups 91 weight(decay=0.0), 98 weight(decay=0.0005),
97 bias(decay=0.0)
TensorBoard: model graph visualization added ✅
Image sizes 640 train, 640 val
Using 8 dataloader workers
Logging results to runs/detect/car_detection_20
Starting training for 20 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       1/20      7.12G      1.543      1.915      1.402          0        640: 100%|██████████| 61/61 [00:09<
00:00,  6.30it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00,  9.58it/s]
                   all        362        241      0.846       0.89      0.924      0.554
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       2/20      7.03G      1.424      1.087      1.416          0        640: 100%|██████████| 61/61 [00:09<
00:00,  6.70it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00,  9.55it/s]
                   all        362        241      0.112      0.606     0.0947     0.0547
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       3/20      7.14G      1.455      1.031      1.476          0        640: 100%|██████████| 61/61 [00:08<
00:00,  6.80it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00, 10.36it/s]
                   all        362        241     0.0382      0.776     0.0357     0.0191
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       4/20      6.99G      1.426     0.9197      1.474          2        640: 100%|██████████| 61/61 [00:08<
00:00,  6.87it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00,  9.43it/s]
                   all        362        241      0.254      0.232     0.0931      0.051
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       5/20      7.14G      1.371      0.826      1.426          3        640: 100%|██████████| 61/61 [00:09<
00:00,  6.65it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00, 10.79it/s]
                   all        362        241      0.854      0.874      0.941      0.579
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       6/20      7.11G      1.365     0.7784      1.408          0        640: 100%|██████████| 61/61 [00:08<
00:00,  6.86it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00,  9.24it/s]
                   all        362        241      0.933      0.922      0.963      0.554
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
```

```
       7/20      7.09G      1.369     0.7968      1.395          1        640: 100%|██████████| 61/61 [00:08<
00:00,  6.82it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00, 10.22it/s]
                   all        362        241      0.936      0.964      0.987      0.632

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       8/20      7.03G      1.282     0.7111      1.342          0        640: 100%|██████████| 61/61 [00:08<
00:00,  6.82it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00,  9.25it/s]
                   all        362        241      0.975      0.952      0.989      0.634

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       9/20      7.14G      1.269     0.7029       1.31          0        640: 100%|██████████| 61/61 [00:08<
00:00,  6.83it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00,  9.81it/s]
                   all        362        241      0.963      0.992       0.99      0.653

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      10/20       7.1G      1.208     0.6465      1.286          0        640: 100%|██████████| 61/61 [00:09<
00:00,  6.65it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00, 10.62it/s]
                   all        362        241      0.955      0.983       0.99      0.654

Closing dataloader mosaic
/blue/eel5934/r.anumula/.conda/envs/art/lib/python3.12/site-packages/torch/utils/data/dataloader.py:617: User
Warning: This DataLoader will create 8 worker processes in total. Our suggested max number of worker in curre
nt system is 2, which is smaller than what this DataLoader is going to create. Please be aware that excessive
worker creation might get DataLoader running slow or even freeze, lower the worker number to avoid potential
slowness/freeze if necessary.
  warnings.warn(
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      11/20       7.1G      1.327     0.7113      1.415          1        640: 100%|██████████| 61/61 [00:10<
00:00,  5.99it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00, 11.18it/s]
                   all        362        241       0.95      0.948      0.988      0.666

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      12/20         7G      1.263     0.7012      1.372          0        640: 100%|██████████| 61/61 [00:08<
00:00,  7.18it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00,  9.62it/s]
                   all        362        241      0.926      0.931      0.976      0.642

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      13/20      7.14G      1.251     0.6748       1.38          2        640: 100%|██████████| 61/61 [00:08<
00:00,  7.18it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00, 10.26it/s]
                   all        362        241      0.971      0.975      0.991      0.657

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      14/20      7.09G      1.237     0.6221      1.394          0        640: 100%|██████████| 61/61 [00:08<
00:00,  7.15it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00, 10.29it/s]
                   all        362        241      0.959      0.946      0.988      0.667

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      15/20      7.08G      1.274     0.6249       1.38          0        640: 100%|██████████| 61/61 [00:08<
00:00,  7.09it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00,  9.52it/s]
                   all        362        241      0.975      0.979      0.993      0.674

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      16/20      7.19G      1.223     0.5891      1.378          0        640: 100%|██████████| 61/61 [00:08<
00:00,  7.19it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00, 10.01it/s]
                   all        362        241       0.96      0.984       0.99      0.671

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      17/20      7.14G       1.21     0.5566      1.337          0        640: 100%|██████████| 61/61 [00:08<
00:00,  7.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00,  9.81it/s]
                   all        362        241      0.971      0.986      0.992      0.689

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      18/20      7.09G      1.176     0.5698      1.331          1        640: 100%|██████████| 61/61 [00:08<
00:00,  7.09it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00, 10.82it/s]
                   all        362        241       0.96      0.992      0.992      0.687

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      19/20       7.1G      1.147      0.528      1.348          1        640: 100%|██████████| 61/61 [00:08<
00:00,  7.07it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00, 11.47it/s]
                   all        362        241      0.971      0.987      0.993      0.701

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      20/20      7.02G      1.136     0.5283      1.287          0        640: 100%|██████████| 61/61 [00:08<
00:00,  7.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00, 10.43it/s]
                   all        362        241      0.972      0.996      0.993      0.703

20 epochs completed in 0.095 hours.
Optimizer stripped from runs/detect/car_detection_20/weights/last.pt, 50.5MB
Optimizer stripped from runs/detect/car_detection_20/weights/best.pt, 50.5MB

Validating runs/detect/car_detection_20/weights/best.pt...
Ultralytics 8.3.39 🚀 Python-3.12.7 torch-2.5.1+cu124 CUDA:0 (NVIDIA A100-SXM4-80GB, 81038MiB)
YOLOv5m summary (fused): 248 layers, 25,045,795 parameters, 0 gradients, 64.0 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1
2/12 [00:01<00:00,  8.90it/s]
                   all        362        241      0.972      0.996      0.993      0.703
Speed: 0.1ms preprocess, 1.0ms inference, 0.0ms loss, 0.6ms postprocess per image
Results saved to runs/detect/car_detection_20
```

In [11]:

```python
# If a best model was found, save its information
if best_model_info:
    model_path = best_model_info['model_path']
    results_csv = best_model_info['results_csv']
    with open(pickle_file, 'wb') as f:
        pickle.dump(best_model_info, f)
```

In [14]:

```python
# If best model info is available, plot learning curves
# Assuming best_model_info is defined and df is loaded correctly
if best_model_info:
    df = pd.read_csv(best_model_info["results_csv"])
    metrics = ['train/box_loss', 'train/cls_loss', 'metrics/precision(B)', 'metrics/recall(B)']
    # Create subplots
    fig, axs = plt.subplots(2, 2, figsize=(15, 15))
    axs = axs.ravel()  # Flatten the 2D array to 1D for easier iteration
    # Loop through each metric and plot it
    for i, metric in enumerate(metrics):
        if metric in df.columns:
            axs[i].plot(df['epoch'], df[metric], label=metric)
            axs[i].set_title(f'{metric} vs Epoch')
            axs[i].set_xlabel('Epoch')
            axs[i].set_ylabel('Value')
            axs[i].legend(loc='upper right')
        else:
            axs[i].axis('off')  # Hide the axis if the metric is missing
    plt.tight_layout()
    # Save figure as PNG
    plt.savefig('learning-curves.png')
    print("Learning curves saved as 'learning-curves.png'")
    # Show the plot
    plt.show()
print(f"Best mAP achieved: {best_map}")
```

```
Learning curves saved as 'learning-curves.png'
<Figure size 1500x1500 with 4 Axes>
Best mAP achieved: 0.97215
```

# 3

To assess how well the model performs without relying solely on labels, you can involve a small group of people for additional validation. Select a small set of test images—say, 50 or 100—and show these images to a few individuals, asking them to mark areas where they see cars. Afterward, compare their annotations with the predictions made by your model. This will give you a clearer picture of how accurate your model's predictions are in real-world scenarios.

For images without cars, you can train the model to recognize empty scenes as well. When testing, if the model doesn't detect any cars in an image, you can verify this by reviewing the image manually or asking your human helpers to confirm.

This approach provides a practical way to gauge your model's effectiveness, especially in cases where some images contain cars and others do not. It acts as a reality check, using human judgment to evaluate how well the model is performing.

In [2]:

```
df = pd.read_csv('./car_detection_dataset/train_bounding_boxes.csv')
image_files = sorted([l for l in os.listdir('./car_detection_dataset/training_images')
                      if l.lower().endswith(('.png', '.jpg', '.jpeg'))
                      and os.path.isfile(os.path.join('./car_detection_dataset/training_images', l))])

train_images, val_images = train_test_split(image_files, test_size=0.2, random_state=42)

os.makedirs('./car_detection_dataset/train_images_with_and_without_cars/images', exist_ok=True)
os.makedirs('./car_detection_dataset/train_images_with_and_without_cars/labels', exist_ok=True)
os.makedirs('./car_detection_dataset/val_images_with_and_without_cars/images', exist_ok=True)
os.makedirs('./car_detection_dataset/val_images_with_and_without_cars/labels', exist_ok=True)
```

In [3]:

```
# Process and move training images
for images in train_images:
    source = os.path.join('./car_detection_dataset/training_images', images)

    # Copy image to the destination folder
    destination = os.path.join( './car_detection_dataset/train_images_with_and_without_cars', images)
    shutil.copy(source, destination)

    # Retrieve bounding box data for the current image
    image_data = df[df['image'] == images]
    label_path = os.path.join('./car_detection_dataset/train_images_with_and_without_cars/labels', f'{Path(i

    # Open the label file and write the bounding box annotations
    with open(label_path, 'w') as lp:
        if not image_data.empty:
            image = cv2.imread(source)
            for _, row in image_data.iterrows():
                # Normalize the bounding box coordinates
                x = (row['xmin'] + row['xmax']) / 2 / image.shape[1]
                y = (row['ymin'] + row['ymax']) / 2 / image.shape[0]
                w = (row['xmax'] - row['xmin']) / image.shape[1]
                h = (row['ymax'] - row['ymin']) / image.shape[0]

                # Write the annotation (label) in YOLO format
                lp.write(f'0 {x} {y} {w} {h}\n')
```

In [4]:

```
# Process and move validation images
for images in val_images:
    source = os.path.join('./car_detection_dataset/training_images', images)
    # Copy image to the destination folder
    destination = os.path.join('./car_detection_dataset/val_images_with_and_without_cars', images)
    shutil.copy(source, destination)

    # Retrieve bounding box data for the current image
    image_data = df[df['image'] == images]
    label_path = os.path.join('./car_detection_dataset/val_images_with_and_without_cars/labels', f'{Path(ima

    # Open the label file and write the bounding box annotations
    with open(label_path, 'w') as lp:
        if not image_data.empty:
            image = cv2.imread(source)
            for _, row in image_data.iterrows():
                # Normalize the bounding box coordinates
                x = (row['xmin'] + row['xmax']) / 2 / image.shape[1]
                y = (row['ymin'] + row['ymax']) / 2 / image.shape[0]
                w = (row['xmax'] - row['xmin']) / image.shape[1]
                h = (row['ymax'] - row['ymin']) / image.shape[0]

                # Write the annotation (label) in YOLO format
                lp.write(f'0 {x} {y} {w} {h}\n')
```

In [5]:

```
data_yaml = {
    'train': './car_detection_dataset/train_images_with_and_without_cars/images',
    'val': './car_detection_dataset/val_images_with_and_without_cars/images',
    'nc': 1,  # Number of classes (in this case, only 'car')
    'names': ['car']
}

# Save the YAML configuration to a file
import yaml
with open('dataset_with_and_without_cars.yaml', 'w') as f:
    yaml.dump(data_yaml, f)
```

In [6]:

```
maxWidth, maxHeight = 0, 0
for name in os.listdir('./car_detection_dataset/training_images/'):
    imagePath = os.path.join('./car_detection_dataset/training_images/', name)
    # Read the image
    image = cv2.imread(imagePath)
    if image is not None:  # Ensure the image was read successfully
        height = image.shape[0]
        width =  image.shape[1] # Get the dimensions
        # Update max width and height if necessary
        if width > maxWidth:
            maxWidth = width
        if height > maxHeight:
            maxHeight = height

# Output the maximum width and height
print(f"Max Width: {maxWidth}, Max Height: {maxHeight}")
```

```
Max Width: 676, Max Height: 380
```

In [7]:

```
model = YOLO('yolov5m.pt')
imgsz = max(32 * round(max(maxWidth, maxHeight) / 32), 640)
model_path = './runs/detect/car_detection_20/weights/best.pt'
```

```
PRO TIP 💡 Replace 'model=yolov5m.pt' with new 'model=yolov5mu.pt'.
YOLOv5 'u' models are trained with https://github.com/ultralytics/ultralytics and feature improved performanc
e vs standard YOLOv5 models trained with https://github.com/ultralytics/yolov5.
```

In [8]:

```
def evaluate_predictions(pred_boxes, true_boxes, image, iou_threshold=0.8, buffer=4):
    cars_identified = 0
    total_cars = len(true_boxes)

    img_h = image.shape[0]
    img_w = image.shape[1]

    for true_box in true_boxes:
        x, y, w, h = true_box
        x1 = int((x - w/2) * img_w)
        y1 = int((y - h/2) * img_h)
        x2 = int((x + w/2) * img_w)
        y2 = int((y + h/2) * img_h)
        true_box_coords = [x1, y1, x2, y2]

        b2 = [x1 - buffer, y1 - buffer, x2 + buffer, y2 + buffer]
        for pred_box in pred_boxes:
            pred_x1, pred_y1, pred_x2, pred_y2 = pred_box

            # Calculate intersection area
            inter_x1 = max(pred_x1, b2[0])
            inter_y1 = max(pred_y1, b2[1])
            inter_x2 = min(pred_x2, b2[2])
            inter_y2 = min(pred_y2, b2[3])

            inter_area = max(0, inter_x2 - inter_x1) * max(0, inter_y2 - inter_y1)
            pred_area = (pred_x2 - pred_x1) * (pred_y2 - pred_y1)
            true_area = (b2[2] - b2[0]) * (b2[3] - b2[1])

            iou = inter_area / float(pred_area + true_area - inter_area)

            if iou > iou_threshold:
                cars_identified += 1
                break
    return cars_identified, total_cars
```

In [10]:

```
def load_true_boxes(label_path):
    """Load ground truth bounding boxes from a label file."""
    try:
        with open(label_path, 'r') as f:
            return [list(map(float, line.strip().split()[1:])) for line in f]
    except FileNotFoundError:
        return []

def evaluate_and_accumulate(image_path, label_path, model, iou_threshold=0.8, buffer=4):
    """Evaluate predictions on a single image and accumulate identified and total car counts."""
    image = cv2.imread(image_path)
    predictions = model(image_path, verbose=False)
    pred_boxes = predictions[0].boxes.xyxy.cpu().numpy()

    true_boxes = load_true_boxes(label_path)

    cars_identified, cars_in_image = evaluate_predictions(pred_boxes, true_boxes, image, iou_threshold, buff

    return cars_identified, cars_in_image

def calculate_accuracy(total_identified, total_actual):
    """Calculate accuracy based on total identified and total actual cars."""
    if total_actual > 0:
        return total_identified / total_actual
    else:
        return 0.0

def main(image_dir, label_dir, model_path, iou_threshold=0.8, buffer=4):
    model = YOLO(model_path)

    total_cars_identified = 0
    total_cars = 0

    for image_file in os.listdir(image_dir):
        if not image_file.lower().endswith(('.png', '.jpg', '.jpeg')):
            continue

        image_path = os.path.join(image_dir, image_file)
        label_path = os.path.join(label_dir, os.path.splitext(image_file)[0] + '.txt')
        cars_identified, cars_in_image = evaluate_and_accumulate(image_path, label_path, model, iou_threshol

        total_cars_identified += cars_identified
        total_cars += cars_in_image

    accuracy = calculate_accuracy(total_cars_identified, total_cars)

    print(f"Cars identified: {total_cars_identified}")
    print(f"Total cars: {total_cars}")
    print(f"Accuracy: {accuracy:.2f}")

# Define directories and model path
image_dir = './Make sense/'
label_dir = './Make sense labels/'
model_path = './runs/detect/car_detection_20/weights/best.pt'

# Run the evaluation
main(image_dir, label_dir, model_path)
```

```
Cars identified: 1
Total cars: 12
Accuracy: 0.08
```

In [ ]: