Vision and Cognitive Systems:

Vehicle detection and counting Using YOLO architecture

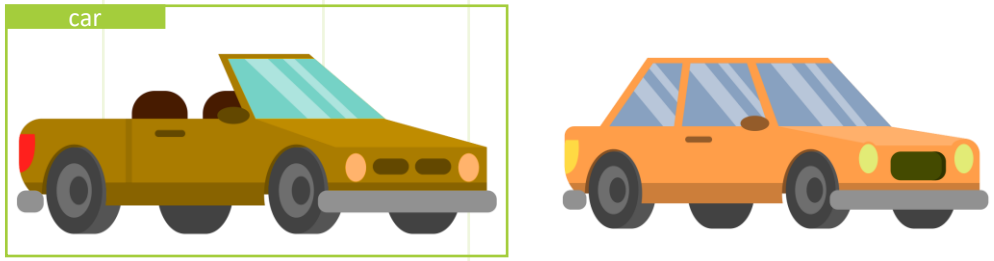
model

members

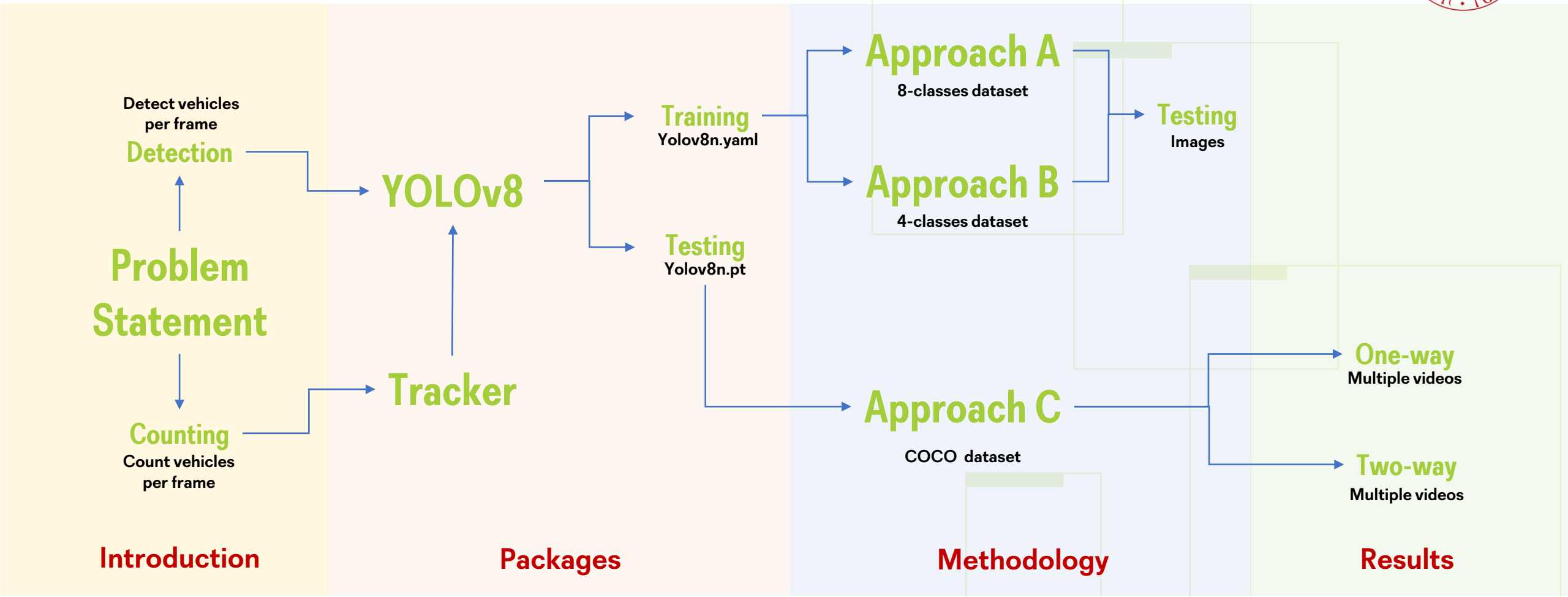
David Alejandro Altamirano Coello

Stefano Rafael Meza Anzules

2023-2024



Outline


Detect vehicles per frame
Detection
Problem Statement
Counting
Count vehicles per frame
YOLOv8
Tracker
Training
Yolov8n.yaml
Testing
Yolov8n.pt
Approach A
8-classes dataset
Approach B
4-classes dataset
Approach C
COCO dataset
Testing
Images
One-way
Multiple videos
Two-way
Multiple videos
Introduction
Packages
Methodology
Results

Introduction

Detecting and Counting

Very important for making informed decisions and ensuring safe navigation

Develop a algorithm capable of **detecting** different objects in a given frame.

Stablish a method capable of **counting** objects in a video.

Distinguish different classes of objects.

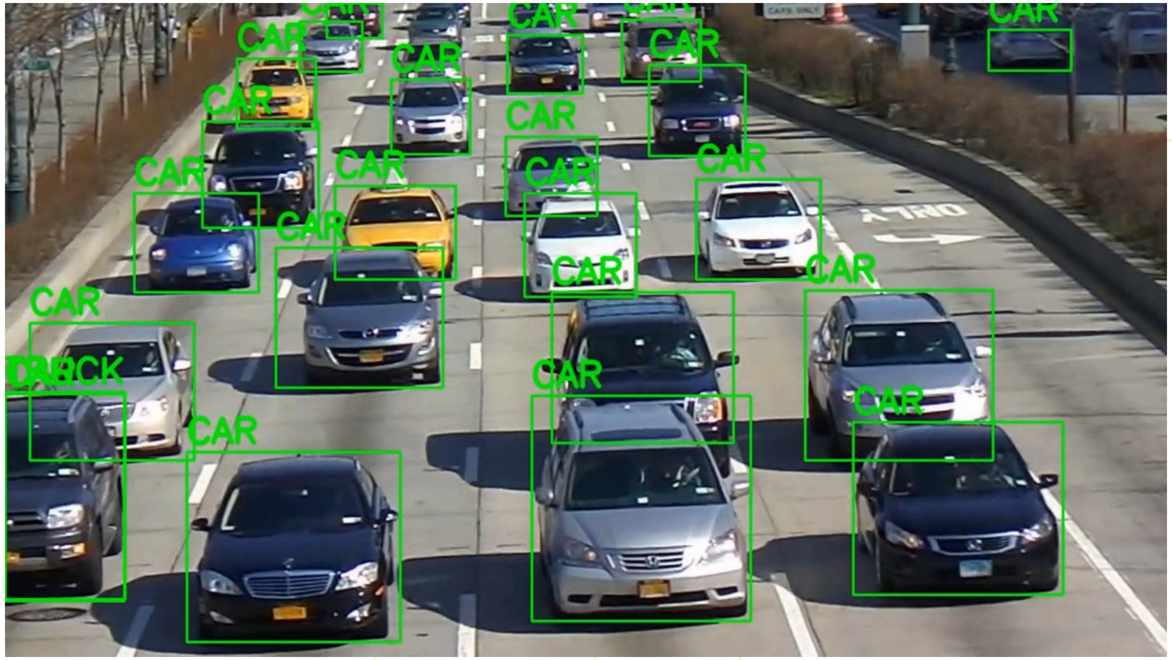


Fig. 1.1: Car detection .

Packages

YOLOv8n

	from	n	params	module	arguments
0	-1	1	464	ultralytics.nn.modules.conv.Conv	[3, 16, 3, 2]
1	-1	1	4672	ultralytics.nn.modules.conv.Conv	[16, 32, 3, 2]
2	-1	1	7360	ultralytics.nn.modules.block.C2f	[32, 32, 1, True]
3	-1	1	18560	ultralytics.nn.modules.conv.Conv	[32, 64, 3, 2]
4	-1	2	49664	ultralytics.nn.modules.block.C2f	[64, 64, 2, True]
5	-1	1	73984	ultralytics.nn.modules.conv.Conv	[64, 128, 3, 2]
6	-1	2	197632	ultralytics.nn.modules.block.C2f	[128, 128, 2, True]
7	-1	1	295424	ultralytics.nn.modules.conv.Conv	[128, 256, 3, 2]
8	-1	1	460288	ultralytics.nn.modules.block.C2f	[256, 256, 1, True]
9	-1	1	164608	ultralytics.nn.modules.block.SPPF	[256, 256, 5]
10	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
11	[-1, 6]	1	0	ultralytics.nn.modules.conv.Concat	[1]
12	-1	1	148224	ultralytics.nn.modules.block.C2f	[384, 128, 1]
13	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
14	[-1, 4]	1	0	ultralytics.nn.modules.conv.Concat	[1]
15	-1	1	37248	ultralytics.nn.modules.block.C2f	[192, 64, 1]
16	-1	1	36992	ultralytics.nn.modules.conv.Conv	[64, 64, 3, 2]
17	[-1, 12]	1	0	ultralytics.nn.modules.conv.Concat	[1]
18	-1	1	123648	ultralytics.nn.modules.block.C2f	[192, 128, 1]

Backbone

Head

layers
225
parameters
3011628

Fig. 2.1: YOLOv8n model .

New features

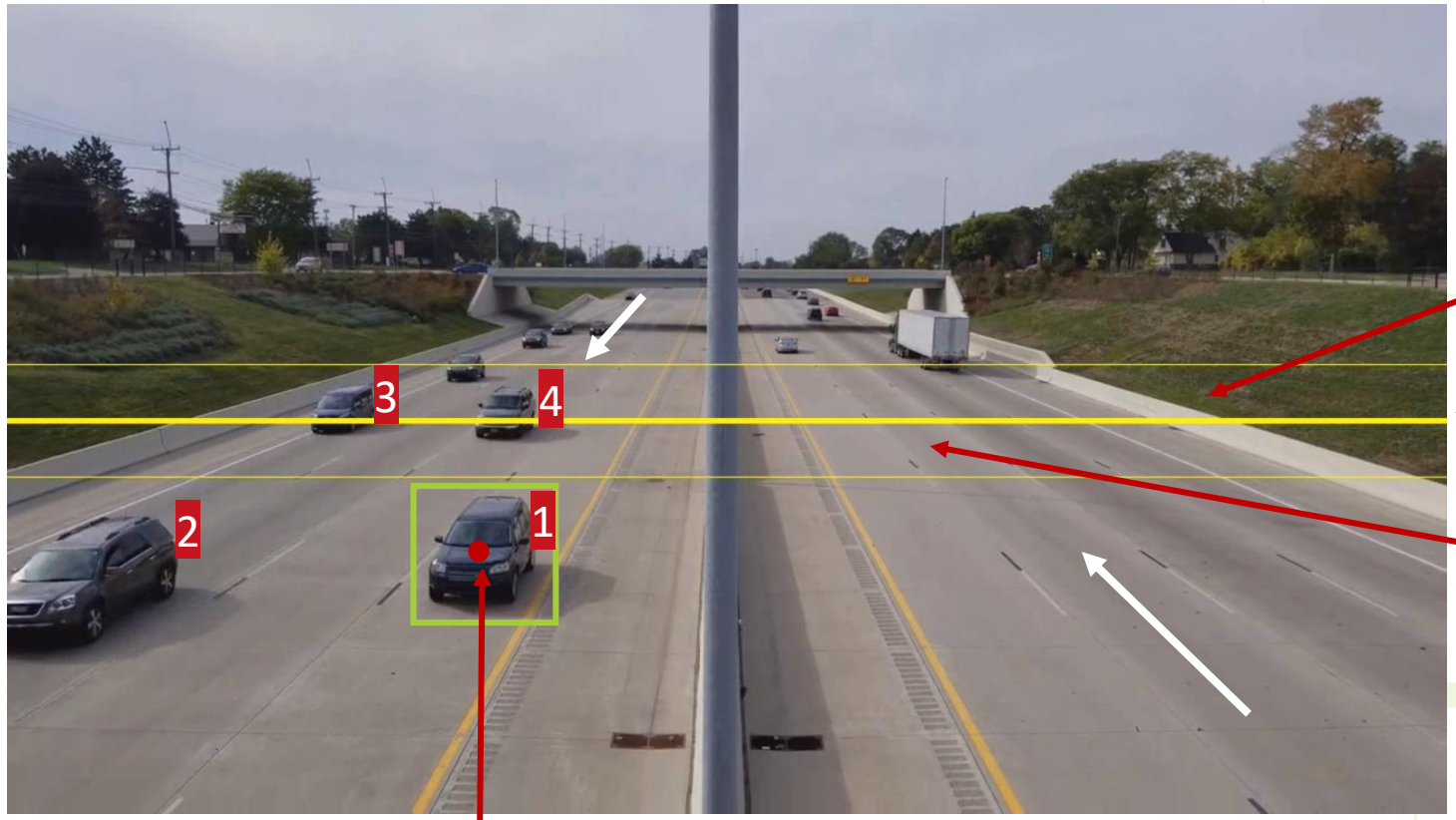
Anchor free detection

New convolution

Mosaic augmentation

Packages

Tracking and counting



Region A (incoming)

Set a line for counting objects

Region B (outcoming)

TRACKER function follows object throughout frames for avoiding double counting

Center point of each object

Approach A

Dataset

8-classes dataset: van, car, SUV, big-truck, truck, bus, motorcycle, pickup
2592 images in total divided in training (70%), validation (20%), test (10%)

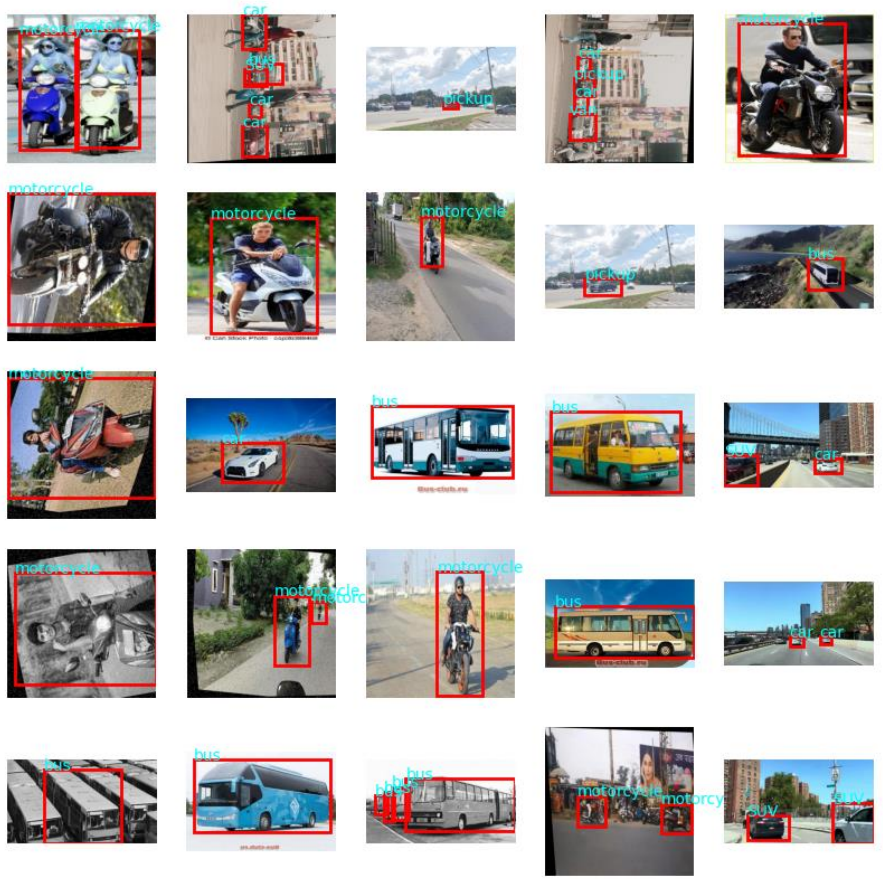


Fig. 3.1: Dataset with corresponding bounding boxes and labels.

Approach A

Training

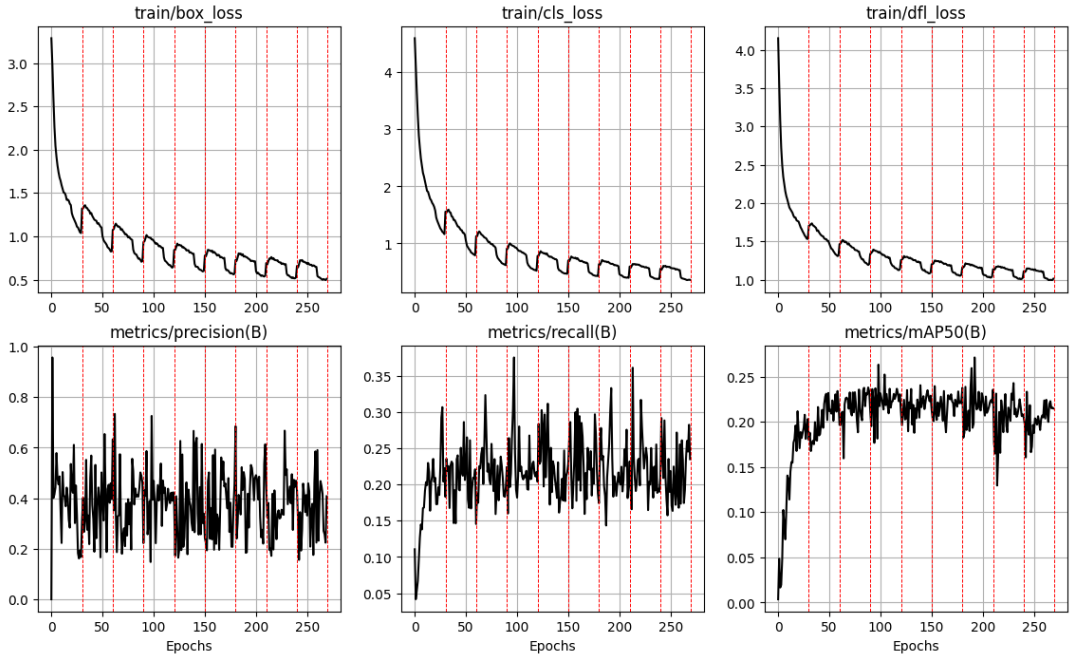


Fig. 3.2: Approch A training parameters using yolov8n.yaml

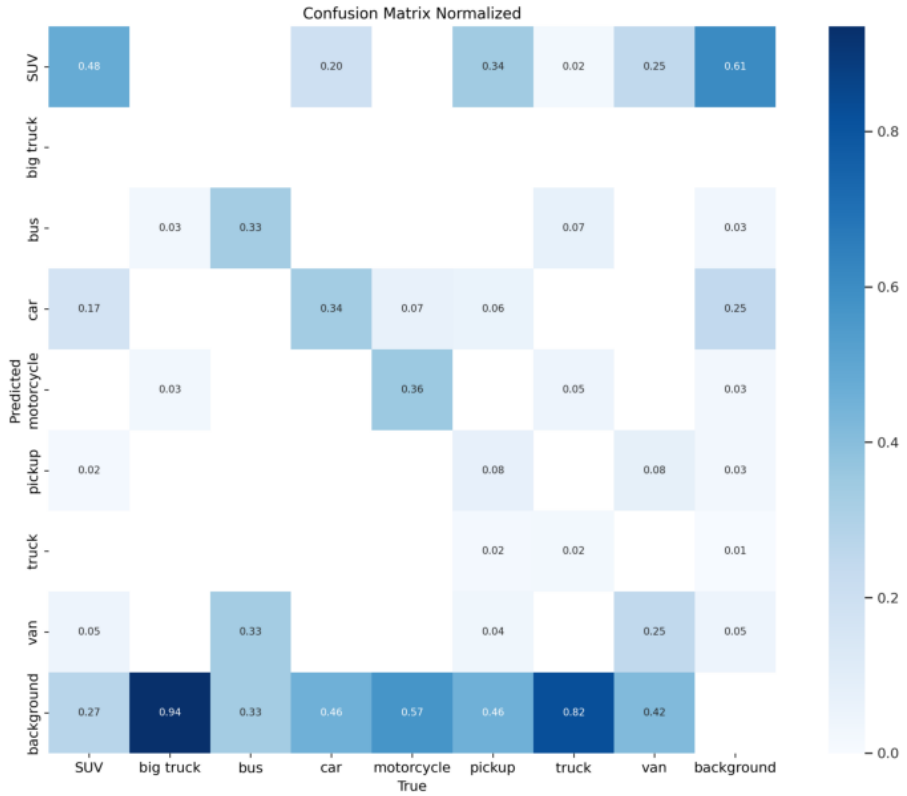


Fig. 3.3: Approach A confussion matrix

Approach A

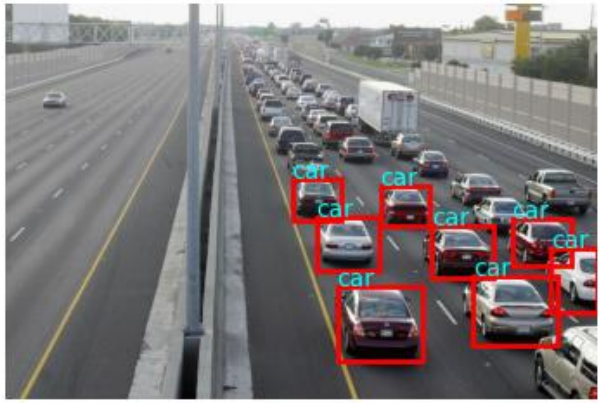
Results

Ex1. Original

Ex1. Validation

Fig. 3.4: Approch A testing with validation set

Num. vehicles: 8



Num. vehicles: 1

Num. vehicles: 3

Fig. 3.6: Approch A testing with test set

Approach B

Dataset

4-classes dataset: car, bus, trucks, motorcycle

cars: 1286 **buses:** 503

trucks: 832 **motorcyles**: 555

3176 images in total

Fig.4.1: Dataset with corresponding bounding boxes and labels.

Approach B

Training

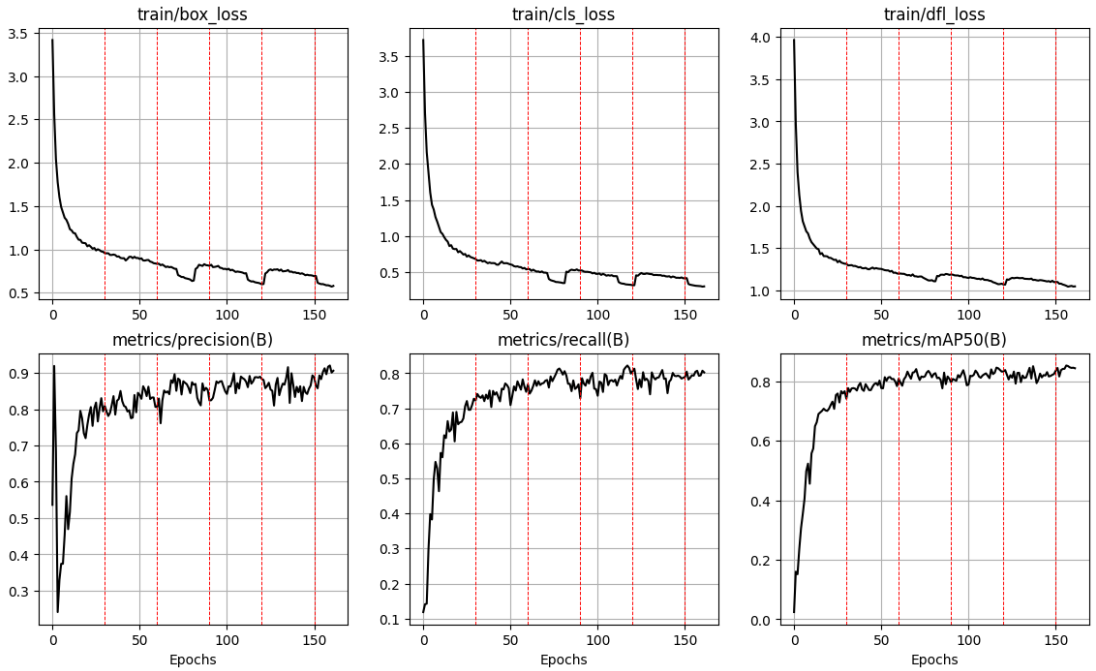


Fig.4.2: Approch B training parameters using yolov8n.yaml

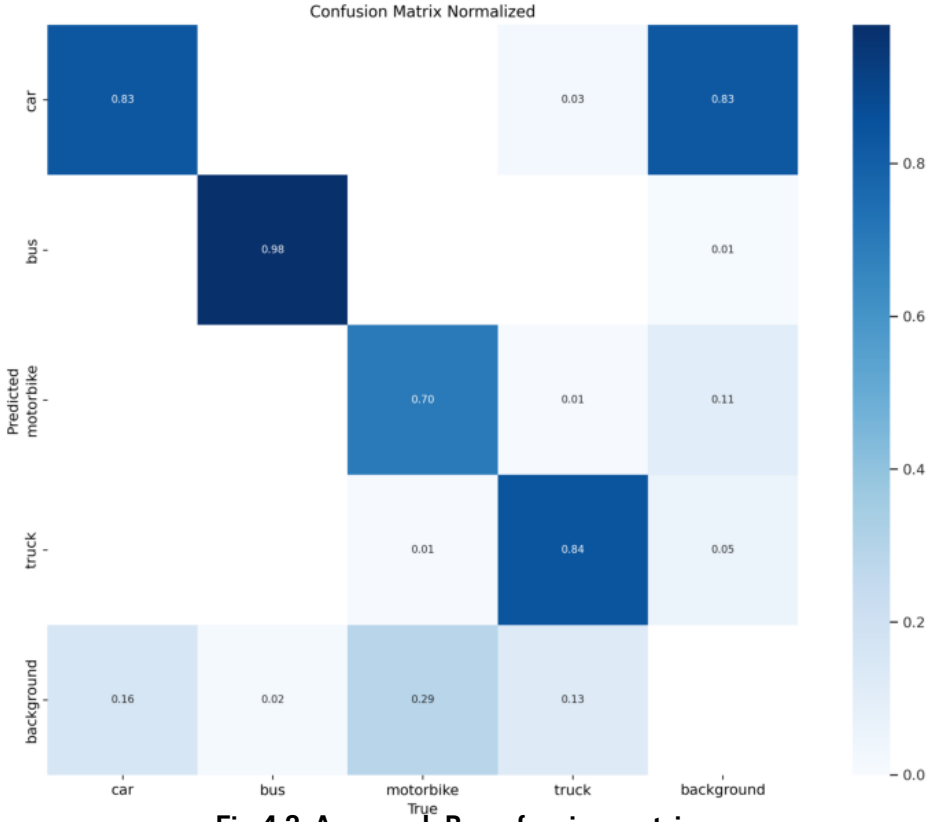


Fig.4.3: Approach B confussion matrix

Approach B

Results

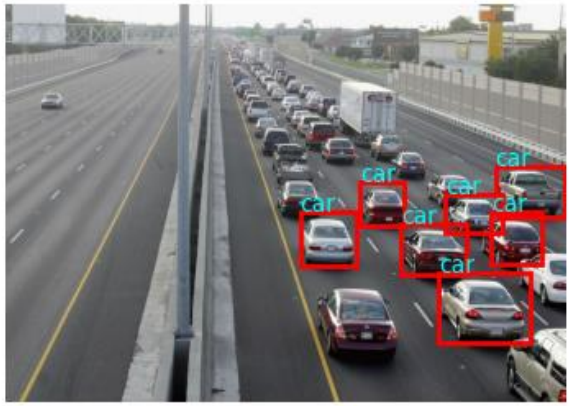
Ex1. Original

Ex1. Validation

Fig. 4.4: Approch B testing with validation set

Num. vehicles: 4

Num. vehicles: 7



Num. vehicles: 7

Fig. 4.6: Approch B testing with test set

Approach C

Coco Dataset

The COCO (Common Objects in Context) dataset is a large-scale image recognition dataset which contains over 330,000 images, each annotated with 80 object categories.
YOLO has a pretrained model for all size versions, here it was used YOLOv8n.pt

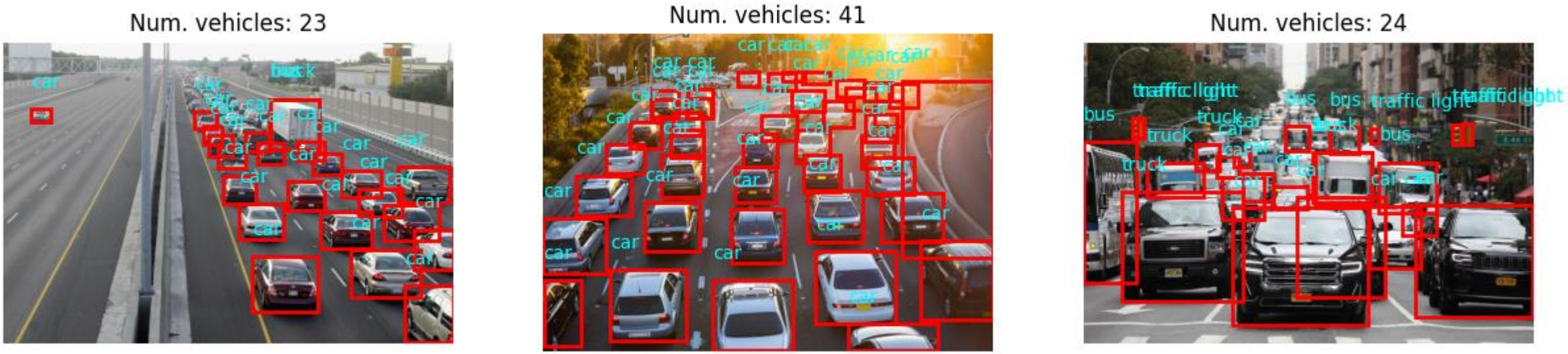


Fig. 5.1: Approch C testing with test set

Results I

One-way

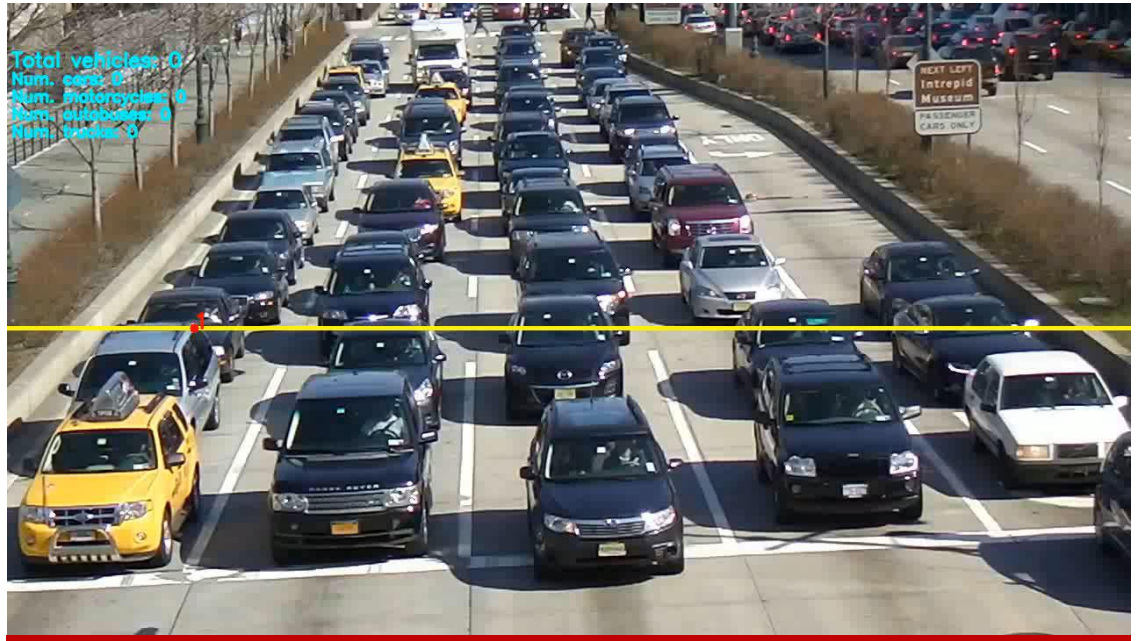
Vehicle detection and counting using pre-trained model (COCO)

4 classes in total without taking into account the direction of the vehicle.

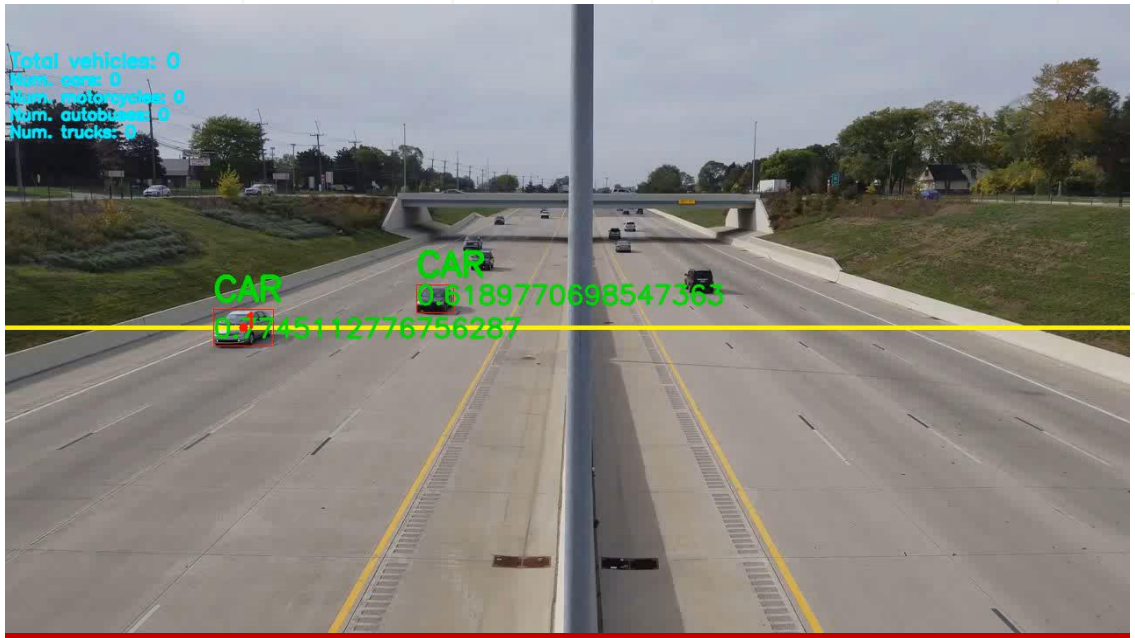
Cars: 29
Motorbikes: 0
Autobuses: 1
Trucks: 6

Results I

One-way



Heavy

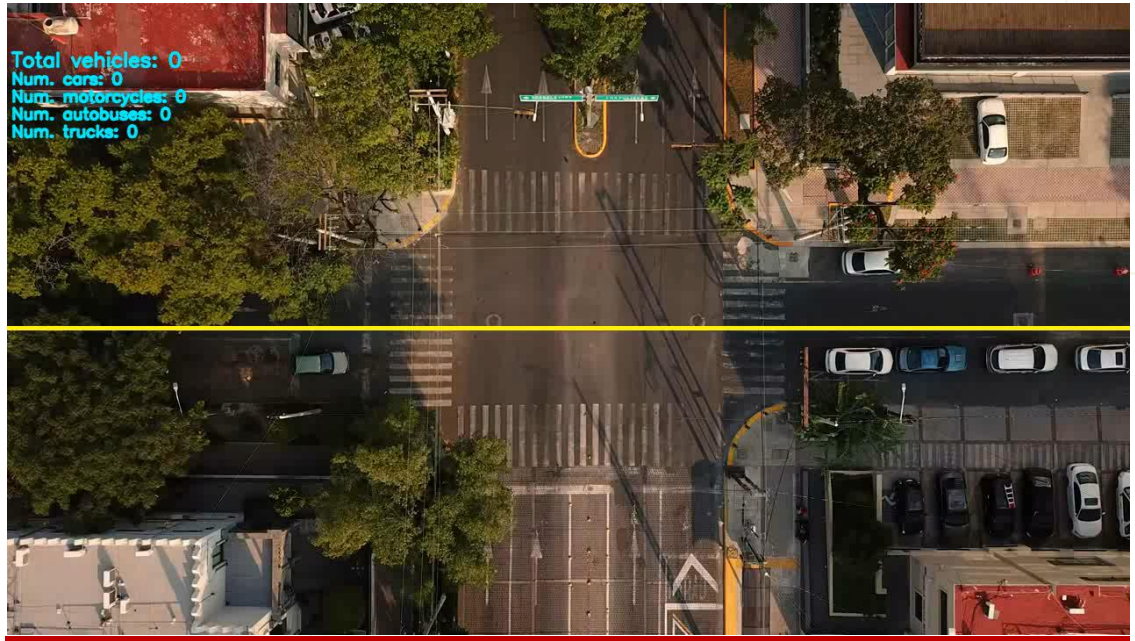


Fast

Results I

One-way

mixed



aerial

Results II

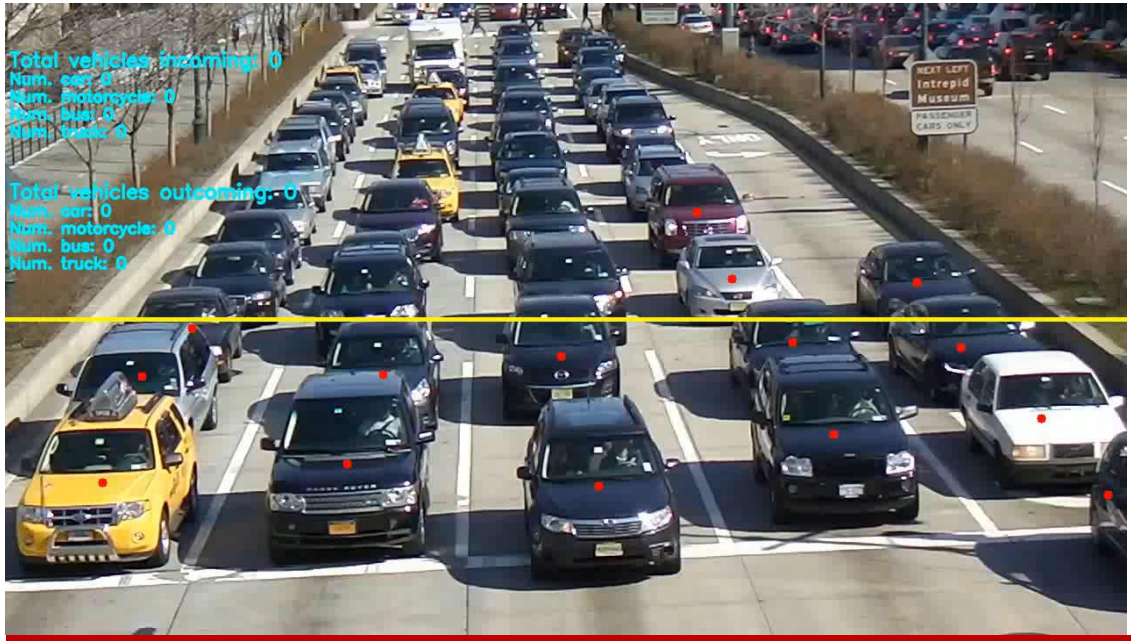
Two-way

Vehicle detection and counting using pre-trained model (COCO)

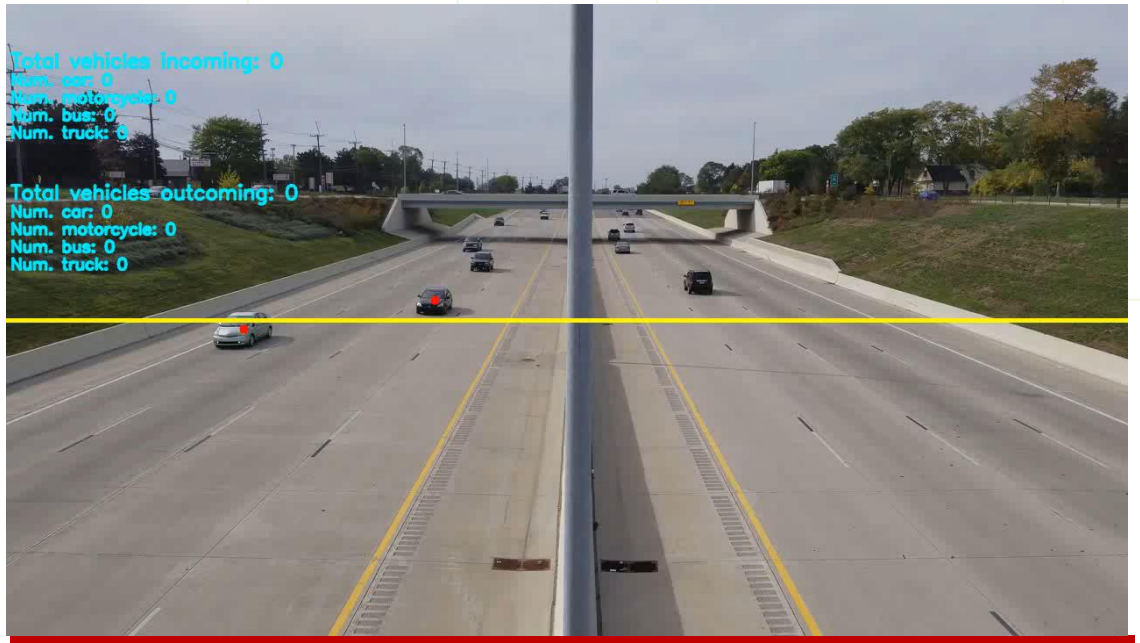
4 classes in total taking into account the direction of the vehicle.

Results II

Two-way



Heavy

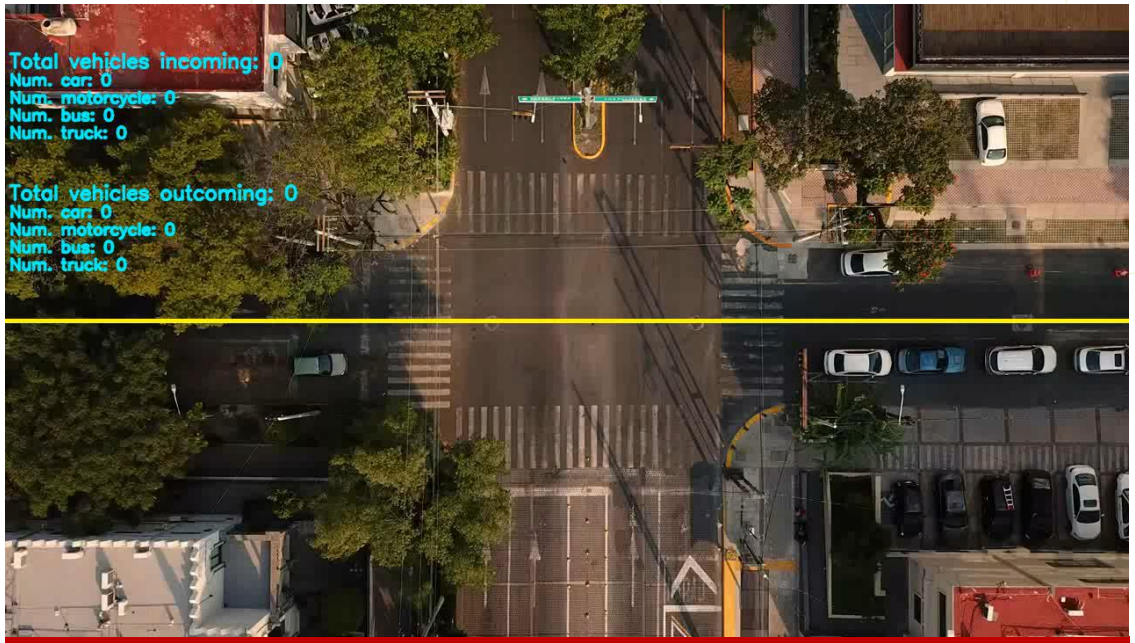


Fast

Results II

Two-way

mixed



aerial

Final remarks

Next updates

- YOLO is one of the best architectures for first timers
- YOLOv8n.pt was the most accurate model for detecting and counting vehicles.
- Aerial recognition did not work properly probably for lack of a more acurate dataset
- Improve dataset and train for more epochs

Suplementary

approach	cars detected	relative error
True	37	
A	12	0.66
B	10	0.72
C	37	0

	cars	motorycles	buses	trucks
heavy	165	0	0	1
fast	234	0	0	18
mixed	82	7	3	17
aerial	0	0	0	0

	direction	cars	motorcycles	buses	trucks
heavy	in	24	0	0	0
	out	114	0	0	1
fast	in	80	0	0	6
	out	64	0	0	6
mixed	in	82	7	3	17
	out	31	0	1	7
aerial	in	0	0	0	0
	out	0	0	0	0