Semantic segmentation for the detection of lanes in a driving simulation environment

Final project

Omar Rodrigo Muñoz Gómez

Outline

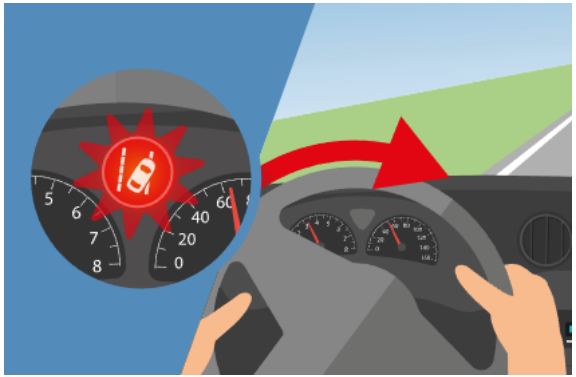
Introduction

Motivation

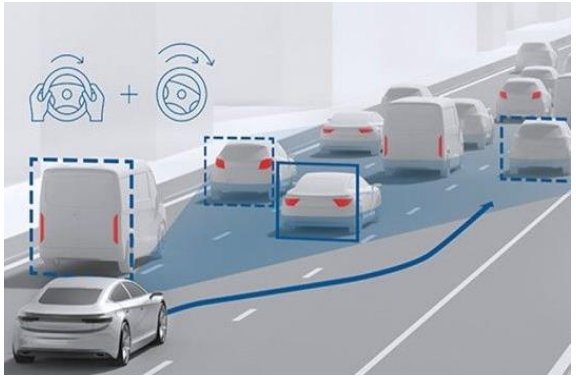
MOTIVATION: WHY LANE DETECTION?

With the help of **sensors**, driver assitant features construct a **model** of their **environment** to help vehicles navigate through it. With such models, we can build different **driving functions**.

LANE DEPARTURE WARNING



LANE KEEP ASSIST

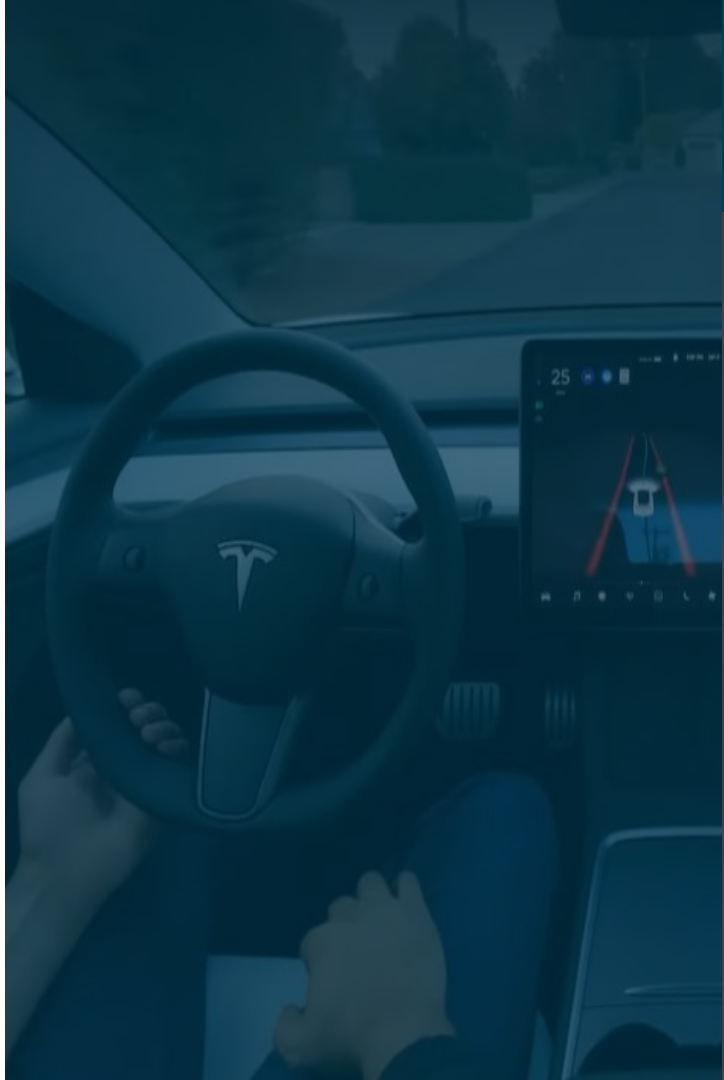


LANE CHANGE ASSIST

MOTIVATION

"The global autonomous vehicle market was valued at $76.13 billion in 2020 and is projected to reach $2,161,79 billion by 2030 [1]."

Thus, the field has had an increased interest in the past few years.

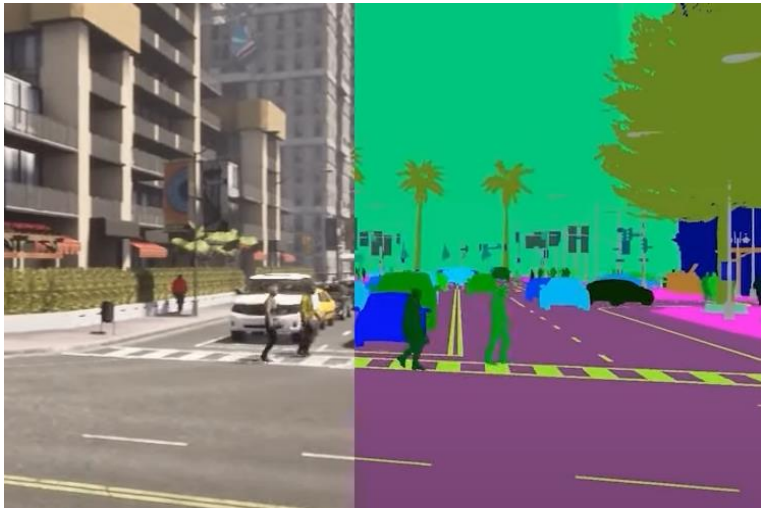


[1] Research and Markets. Autonomous Vehicle Market by Level of Automation, Application, Drive Type, and Vehicle Type: Global Opportunity Analysis and Industry Forecast, 2021-2030. https://www.researchandmarkets.com/reports/5578161/autonomous-vehicle-market-by-level-of-automation, March 2022.

MOTIVATION: WHY USING SIMULATION?

Autonomous vehicles need thousands of training data to provide proper predictions.

Companies are increasingly making use of simulation environments to gather training data and to build corner cases.



We can get perfect lables for free!

CARLA SIMULATION ENVIRONMENT

CARLA is an open-source simulator for autonomous driving research.

It features:

- **C++ and Python API** to control all simulation aspects: traffic, pedestrians, weather, sensors, and more.
- **Autonomous driving sensor suite:** LIDARs, cameras, depth sensors, GPS, among others.
- **Fast simulation** for planning and control.
- **Map** generation.
- **Traffic** simulation

Objective

Perform real time lane detection for a controlled simulation environment in CARLA.

Methodology

Approach

Model generation

CARLA

- Load virtual map
- Load test vehicle
- Define RGB camera
- Create label image
- Store raw video and labels

→ Image and label data →

Python

- Divide train and test
- Image preprocessing
- Train UNET model
- Save model

Model deployment

CARLA

- Load virtual map
- Load test vehicle
- Define RGB camera

→ Image →

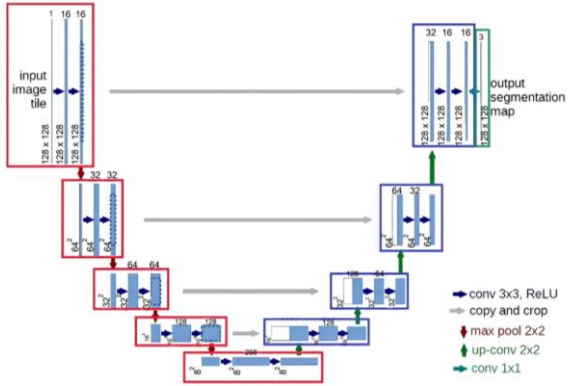
Python

- Load model
- Image preprocessing
- Predict and overlay

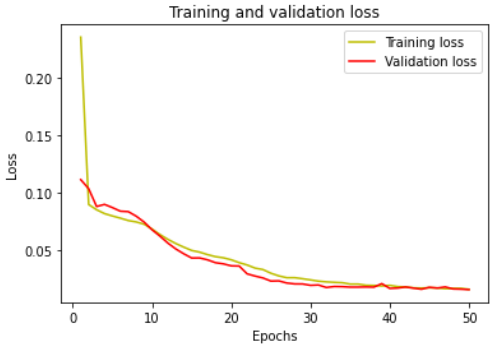
UNET MODEL

Trainable parameters:
1,940,851

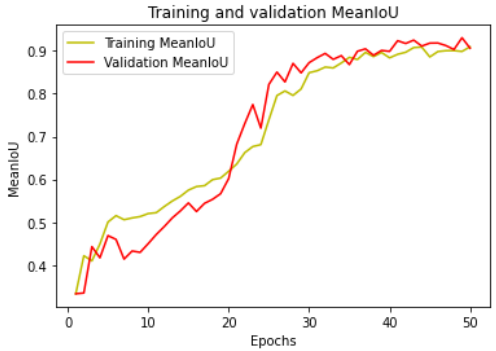
Dataset: 476 images
Training = 354
Test = 118



PERFORMANCE RESULTS

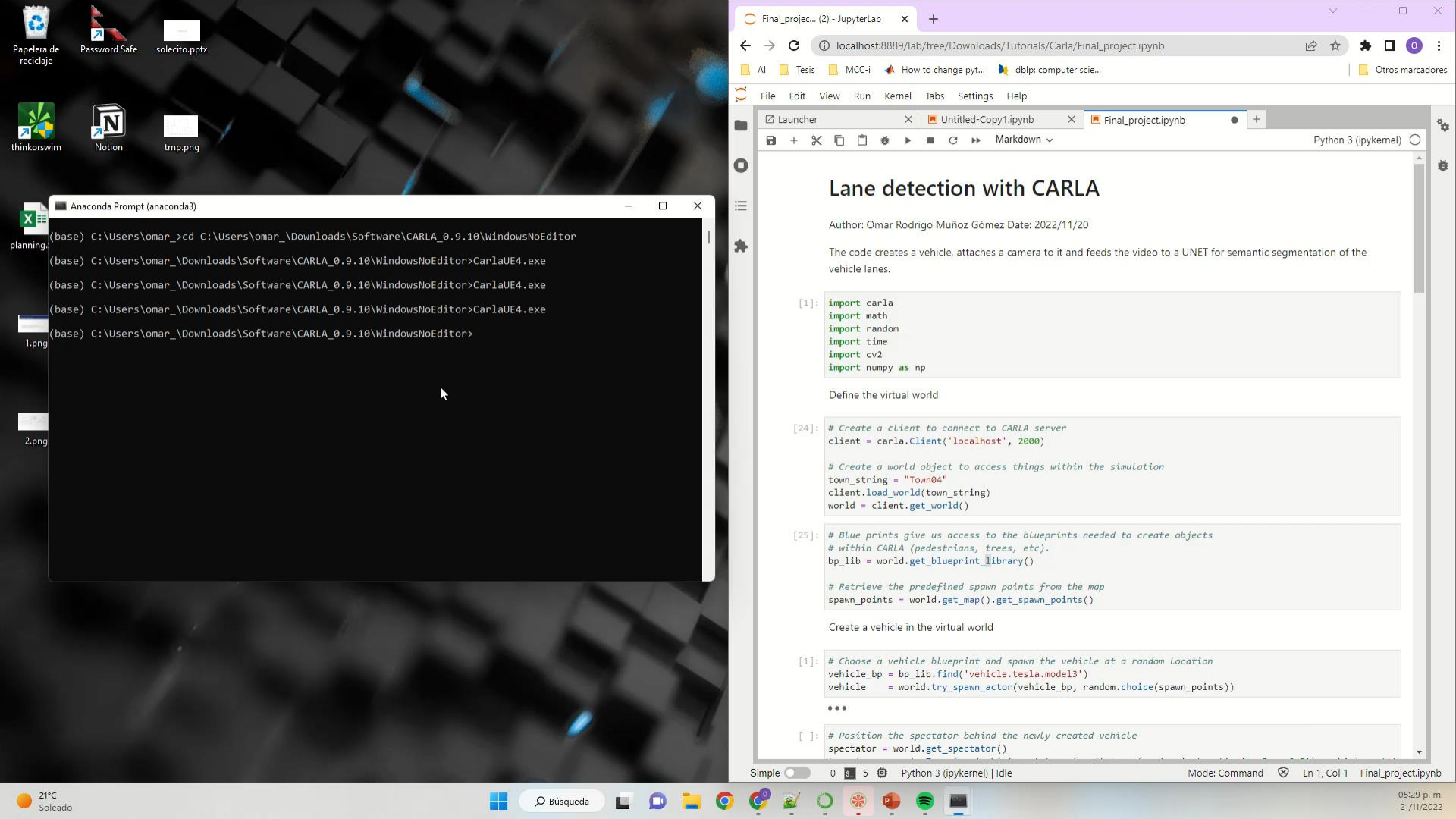


Categorical cross-entropy



Intersection over union

DEMO


Papelera de reciclaje
Password Safe
solecito.pptx
thinkorswim
Notion
tmp.png
planning.
1.png
2.png
Anaconda Prompt (anaconda3)
(base) C:\Users\omar_>cd C:\Users\omar_\Downloads\Software\CARLA_0.9.10\WindowsNoEditor
(base) C:\Users\omar_\Downloads\Software\CARLA_0.9.10\WindowsNoEditor>CarlaUE4.exe
(base) C:\Users\omar_\Downloads\Software\CARLA_0.9.10\WindowsNoEditor>CarlaUE4.exe
(base) C:\Users\omar_\Downloads\Software\CARLA_0.9.10\WindowsNoEditor>CarlaUE4.exe
(base) C:\Users\omar_\Downloads\Software\CARLA_0.9.10\WindowsNoEditor>
Final_projec... (2) - JupyterLab
localhost:8889/lab/tree/Downloads/Tutorials/Carla/Final_project.ipynb
AI
Tesis
MCC-i
How to change pyt...
dblp: computer scie...
Otros marcadores
File
Edit
View
Run
Kernel
Tabs
Settings
Help
Launcher
Untitled-Copy1.ipynb
Final_project.ipynb
Markdown
Python 3 (ipykernel)
Lane detection with CARLA
Author: Omar Rodrigo Muñoz Gómez Date: 2022/11/20
The code creates a vehicle, attaches a camera to it and feeds the video to a UNET for semantic segmentation of the vehicle lanes.
[1]:
import carla
import math
import random
import time
import cv2
import numpy as np
Define the virtual world
[24]:
# Create a client to connect to CARLA server
client = carla.Client('localhost', 2000)
# Create a world object to access things within the simulation
town_string = "Town04"
client.load_world(town_string)
world = client.get_world()
[25]:
# Blue prints give us access to the blueprints needed to create objects
# within CARLA (pedestrians, trees, etc).
bp_lib = world.get_blueprint_library()
# Retrieve the predefined spawn points from the map
spawn_points = world.get_map().get_spawn_points()
Create a vehicle in the virtual world
[1]:
# Choose a vehicle blueprint and spawn the vehicle at a random location
vehicle_bp = bp_lib.find('vehicle.tesla.model3')
vehicle   = world.try_spawn_actor(vehicle_bp, random.choice(spawn_points))
[ ]:
# Position the spectator behind the newly created vehicle
spectator = world.get_spectator()
Simple
0
5
Python 3 (ipykernel) | Idle
Mode: Command
Ln 1, Col 1
Final_project.ipynb
21°C
Soleado
Búsqueda
05:29 p. m.
21/11/2022

Conclusions and future work

- Simulation environment can provide a fast way to generate training data this can result into faster development cycles and testing of corner cases.
- A larger data set might be needed to better generalize. Data augmentation could also lead to better results.
- It seems feasible to perform real time predictions using a small image and a small UNET architecture. The current approach can predict around 20 frames per second.
- To make this approach able to work with real world images, we might need to fine tune the network based on real world images and/or use higher quality simulation images.