Team 14

Implementing a DRL Model for Autonomous Driving

Lesli, Luis De La Mora, Angel Reyes

Background & Motivation

What is DRL?

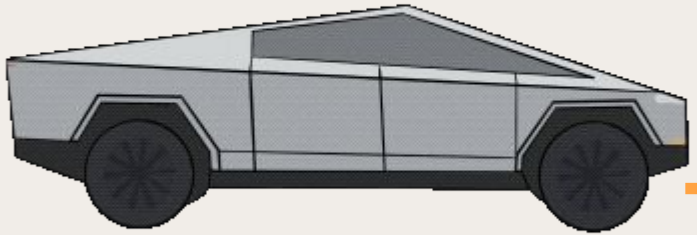
- Subfield of ML, RL + DL
- Makes agent learn optimal actions by interacting with an environment and getting maximum reward.

Motivation

- Minimize road accidents.
- Lower traffic congestion.

CARLA (Car Learning to Act)

- Virtual simulation platform for autonomous driving algorithms.
- Realistic urban environment with 3D models, dynamic traffic, and weather variations.
- Customizable parameters include traffic density, pedestrian behavior, and road layouts.
- Supports integration with Python and C++, accessible to diverse developers.



Hierarchy of Levels of Automation

Level 0	No Automation
Level 1	Driver Assistance
Level 2	Partial Automation
Level 3	Conditional Automation
Level 4	High Autonomy
Level 5	Full Automation

Components of an AV

Planning & Control

1. Throttle
2. Speed
3. Steering
4. Brake (not used)

Car must be able to drive without hitting other cars, bikes, people, objects, etc.

Mapping & Localization

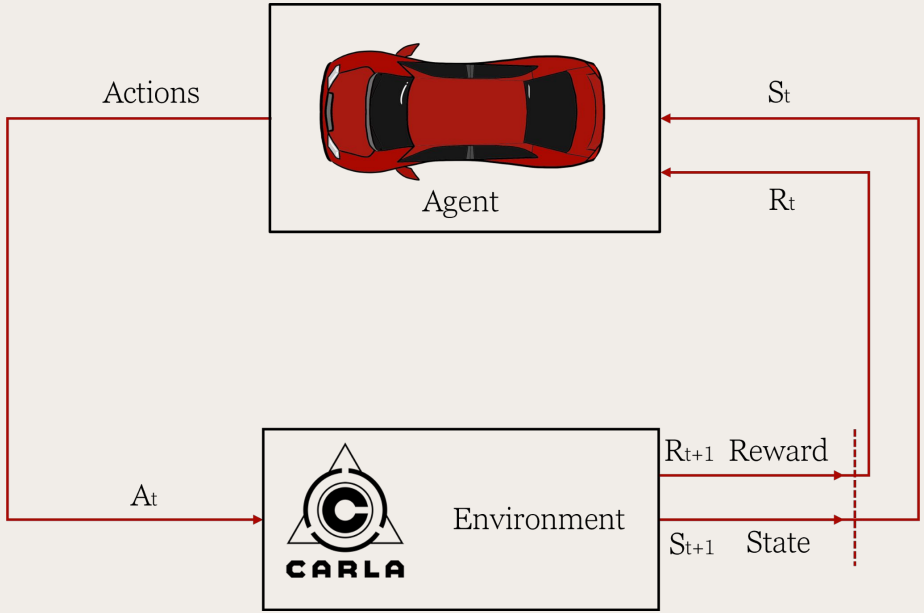
1. Locate vehicle on a map
2. Assess topography

Usage of GPS, environmental data, RGB cameras, etc can help with lane, sign, and object detection.

Simulators

1. Test & validate algorithms before commercializing
2. Must accurately represent physics and environment of the real-world.

Training the Model



Challenges in Training

- Many parameters to change.
- Lengthy training time.
- Optimizing reward function, steering, and throttle.
- Steering lock for early dropping.
- Changing simulation environment for better collision detection.
- Episode length, goal location, spawn location, etc.

Proximal Policy Optimization (PPO)

The Proximal Policy Optimization algorithm combines ideas from A2C (having multiple workers) and TRPO (it uses a trust region to improve the actor).

The main idea is that after an update, the new policy should be not too far from the old policy. For that, ppo uses clipping to avoid too large update.

From SB3 documentation, used in training the model.
Inputs: Environment, Action Space, Learning Rate
Environment is received from SEM Camera located on top of the vehicle.

Simplified Town 6 Training Locations

Simplified Town 6 Training Locations

Simplified Town 6 Training Locations

Spawn Point

Goal

Environment

```
# adding code for GOAL LOCATIONS
self.spawn_locations = [
    carla.Location(x=125.32894897460938, y=244.6654815673828, z=2),
    carla.Location(x=-211.31040954589844, y=-19.262563705444336, z=2),
    carla.Location(x=10.5, y=-50.39863586425781, z=2),
    carla.Location(x=-138.10647583007812, y=239.58534240722656, z=2)
]

self.goal_locations = [
    carla.Location(x=665.9683227539062, y=166.56344604492188, z=2),
    carla.Location(x=-250.22171020507812, y=245.9504852294922, z=2),
    carla.Location(x=-9.634696006774902, y=-34.35064697265625, z=2),
    carla.Location(x=-7, y=240, z=2)
]
```

Originally, 400+ possible spawn locations & small training time.

Environment

```
# reward for making distance
if distance_travelled < 5:
    reward = reward - 10
elif distance_travelled < 10:
    reward = reward - 5
elif distance_travelled < 15:
    reward = reward - 3
elif distance_travelled < 30:
    reward = reward - 1
elif distance_travelled < 50:
    reward =  reward + 1
else:
    reward = reward + 5
```

More distance = Better Reward

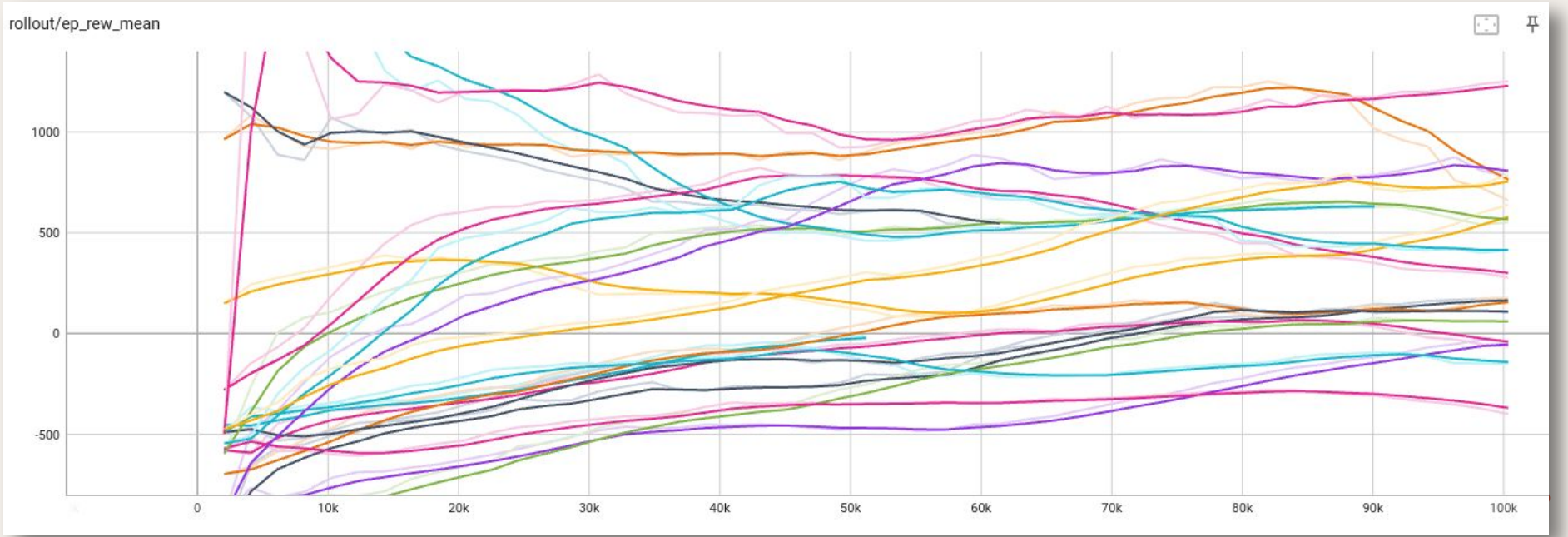
Environment

```
# reward for going near goal
reward = reward + 0.01 * (distance_to_goal - self.prev_distance_to_goal)
self.prev_distance_to_goal = distance_to_goal

# check if the goal is reached
if distance_to_goal < 15:
    reward = reward + 1000  # big reward for reaching the goal
    print("GOAL REACHED!!!!")
    done = True
    self.cleanup()

# check for episode duration
if self.episode_start + SECONDS_PER_EPISODE < time.time():
    print("Episode Length reached")
    done = True
    self.cleanup()
```

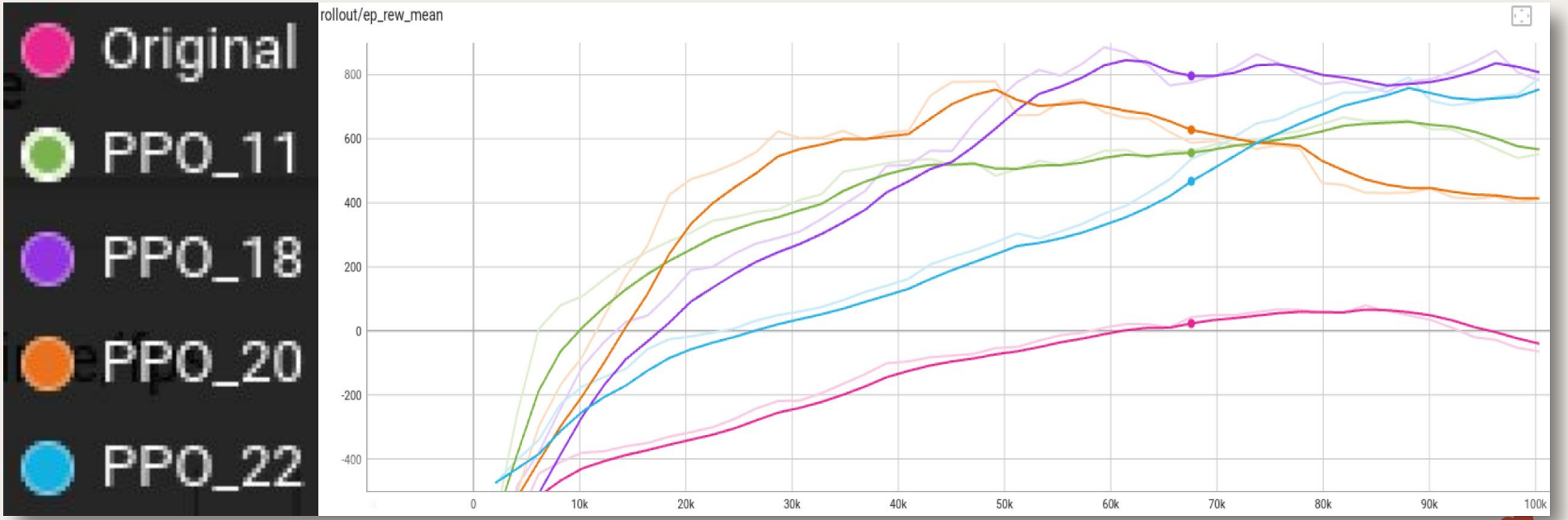
Get vehicle to reach it’s goal by rewarding a small amount as it gets closer.

Results!

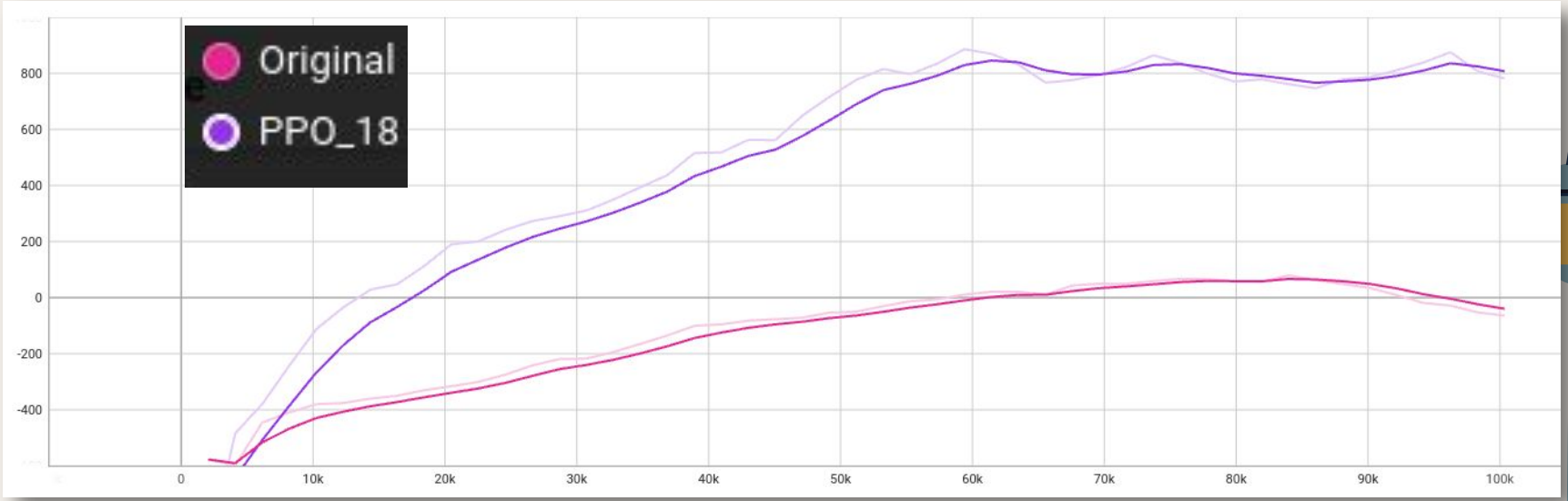
Reward: All of the Models



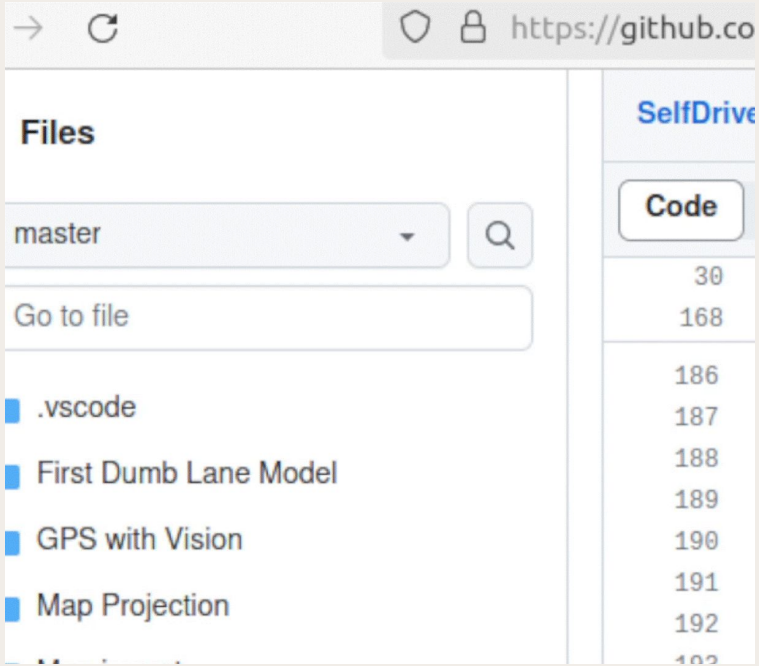
Reward: Original VS Ours



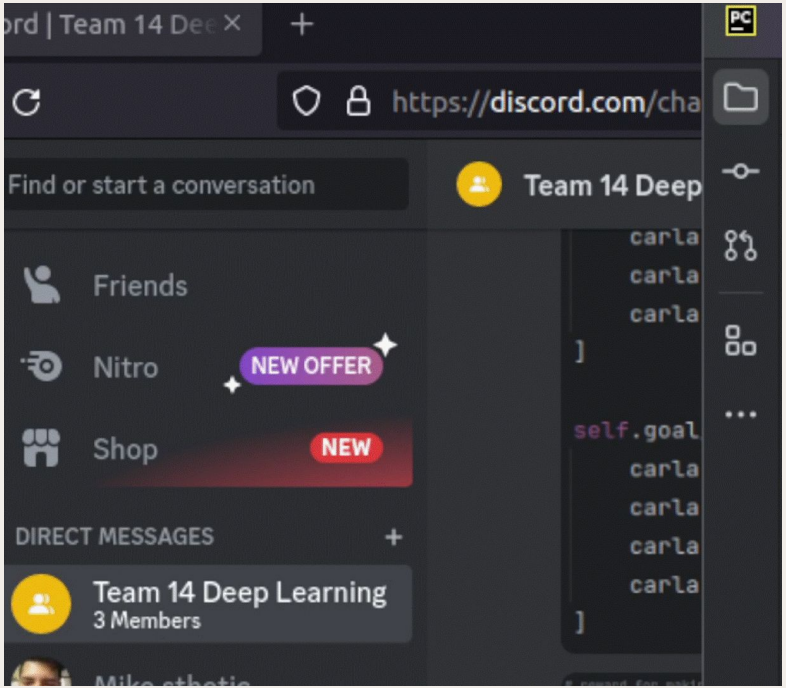
Reward: Original VS Final



Testing Original VS Final Model



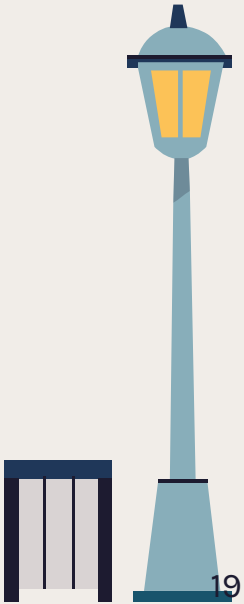
Original Model



Final Model (18)

Changes Made to Create Model 18

- Longer episode length before stopping.
- Training on 4 specific spawn points instead of all available ones.
- Added goals for each spawn point.
- Modified map to be bare.
- Modified map to end episode faster by adding bumpers.
- Added reward function to get reward for achieving goal and getting closer to it.
- Changed negative reward for collisions and steer lock to be more severe.
- Changed amount of time allowed for the car to steer lock before stopping episode.
- Allowed vehicle to be slightly further from goal and still count as hitting it.
- Changed and added accelerator values.



Future Work

Train to stay within road lines.

Include more parameters: breaking, maps, **cars**, etc.

Train multiple models, ex: specific for steering.

Add checkpoints to environment.

https://github.com/vadim7s/SelfDrive/tree/master/RL

Thank you!

Any questions?