Exploring Deep RL Algorithms

Salil Dabholkar
50321748

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Motivation

- Most of the current research in RL is restricted to applications in highly constrained environments like games
- These environments have specific rules and structure which make it easy to design states, rewards, etc.
- But what about actual real world scenarios?
- They are often much messy than games, and have several edge cases which can't be explicitly modelled.
- I wanted to see how RL algorithms perform in such a real-world scenario

- Thus, my primary motivation and purpose for this project was to create a real-world scenario, explore how it could be formulated as a RL problem, and attempt to solve it using a basic algorithm.

Autonomous Driving

- DARPA Urban Challenge remains the largest demonstration of autonomous vehicle technology
- But it excludes many capabilities and requirements critical for actual driving in cities (like pedestrians,bicyclists, traffic)
- Autonomous Driving is a huge space and consists of multiple scenarios (signal following, sign detection, lane following, etc).
- There have been attempts to solve it end-to-end using Deep Learning but training requires a lot of data, computation and time.

- Also most (current) methods approach it as a “single actor problem“ where only one car is being trained.
- But if we are going autonomous, why not have them communicate?

Connected Autonomous Driving

- That’s where Connected Autonomous Driving (CAD) comes into picture.
- Connected Autonomous Vehicles utilize communication systems to improve transportation by enabling cooperative functionalities.
  - It has the ability to share and fuse information gathered from vehicle sensors to create a better understanding of the surrounding.
  - It also enables groups of vehicles to drive in a coordinated way which results in a safer and more efficient driving.

- However, currently there is still a gap in understanding how and to what extent connectivity can contribute in improving the efficiency, safety and performance of autonomous vehicles.

My Environment

- 3 -way stop sign controlled intersection
- Front-view on top-left
- Normally, one car goes while the rest stop

- Is that optimal?
- What if the car at the bottom wanted to turn right?
- Does it need to wait?

- What if the cars could communicate?

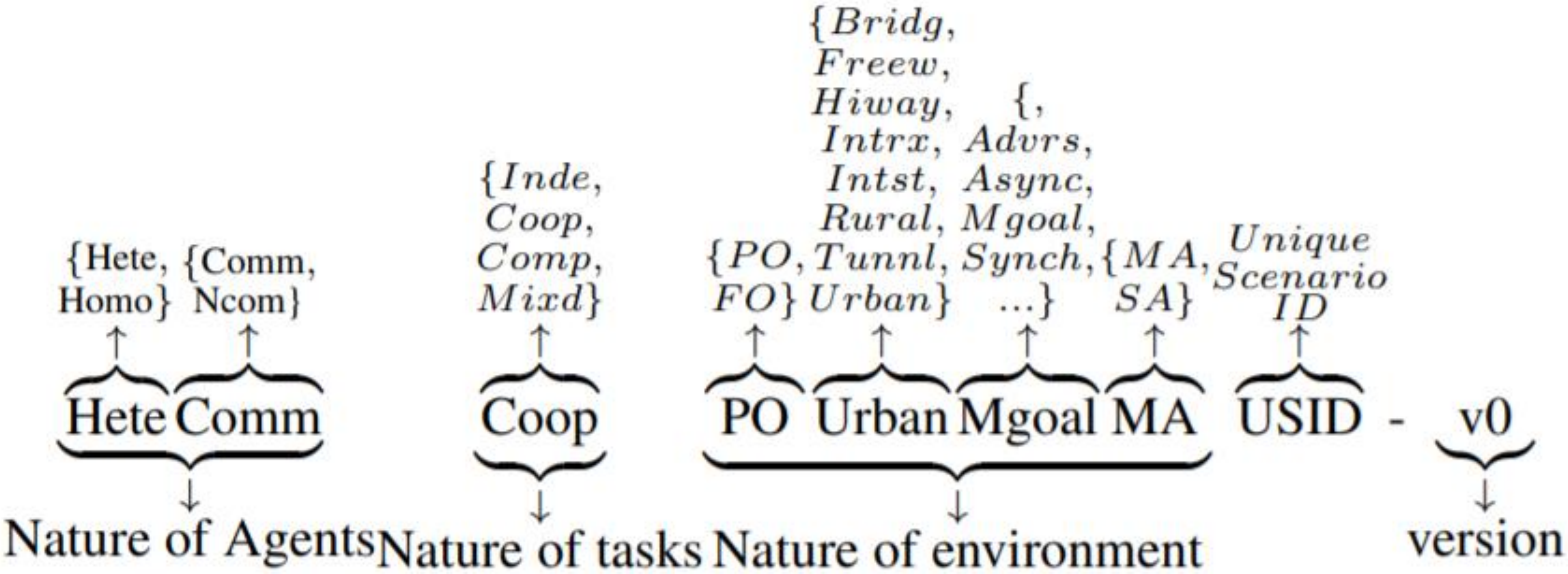
More details

- The environment was created using macad-gym which is like a gym wrapper over the CARLA (Car Learning to Act) simulator.

- I used the three way intersection scenario where the bottom car is trying to turn right, top car going straight, and right car turning left.

- The primary aim was to **avoid collisions** and secondary aim was to get the cars to **cross as quickly as possible**.

About Macad-gym taxonomy



Environment Description

- **Observation space:** The observation for each agent is a 168*168*3 RGB image captured from the camera mounted on the car

- **Action space:** 9 Discrete actions:
    i. Accelerate
    ii. Brake
    iii. Right
    iv. Left
    v. Acc. Left
    vi. Acc. Right
    vii. Brake Right
    viii. Brake Left
    ix. Coast

Rewards

The rewards come from CARLA itself and they were defined as follows:

$$1000\,(D_{t-1} - 1 - D_t) + 0.05\,(V_t - V_{t-1}) - 0.00002\,(C_t - C_{t-1}) - 2(SW_t - SW_{t-1}) - 2\,(OL_t OL_{t-1})$$

- D: distance traveled towards the goal D in km
- V: speed in km/h
- C: collision damage
- SW: intersection with sidewalk
- OL: intersection with opposite lane

Connectivity

- **Shared Parameters:** parameters of each agent's policy can be shared
- **Shared Observations:** Reduces the gap between the observation and the true state
- **Shared Experiences:** This enables collective experience replay which can theoretically lead to gains in a way similar to distributed experience replay.
- **Shared policy:** If all the vehicles follow the same policy π, it follows that the learning objective for each of the agents can be simplified

Setting up Project

The project was setup and executed on UB CCR and new conda environment

- Installing CARLA
  - I used CARLA 0.9.4 – https://carla.readthedocs.io/en/latest/download/
  - Extracted all in ~/software/
  - set the CARLA_SERVER environment variable

- Following Python packages were used:
  - tensorflow 1.14, tensorboard 1.14, ray 0.6.4, macad-gym 0.1.2

- Setup the env in using a json like config file
- Create an agent and execute it normally like with any other gym environment

Experiments

- After setting it up, I solved the environment using basic Policy Gradient algorithm using 2 values of γ: 0.7 and 0.9

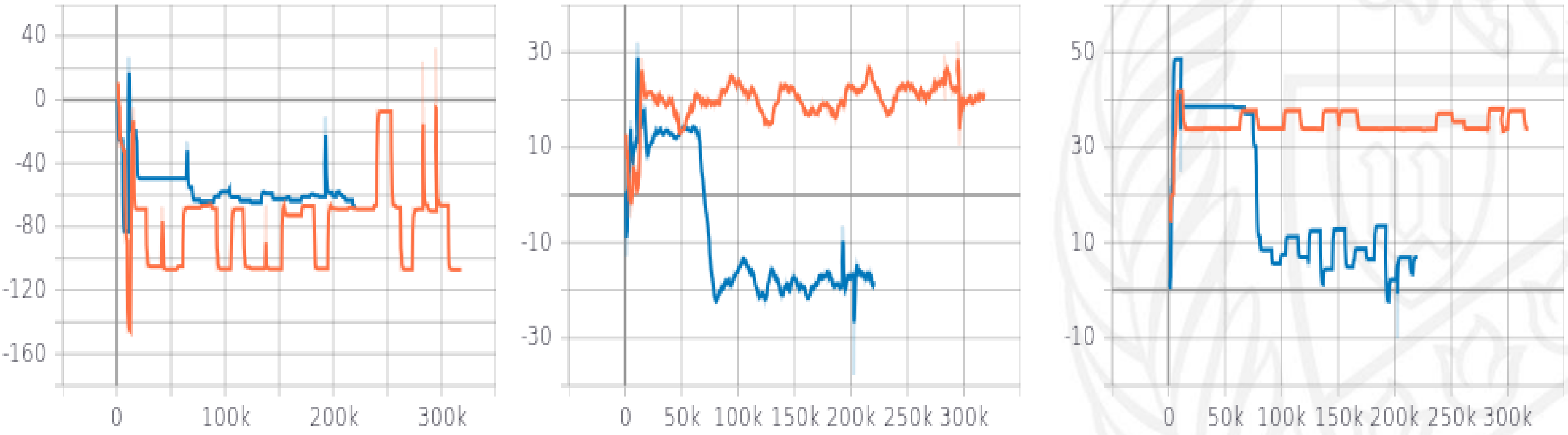
- I made a detailed study of these two experiments on factors like rewards earned, episodes, processing time, and resource consumption

- Training on server and then getting a screenshot is tricky but I managed to capture one good working example

Results - Rewards

γ = 0.7 in orange and γ = 0.9 in blue

γ = 0.7 is always better and monotonous, indicating a stable training performance. A thing of note is that the min reward for γ = 0.9 is always better, indicating that its performance is always better in the worst case.

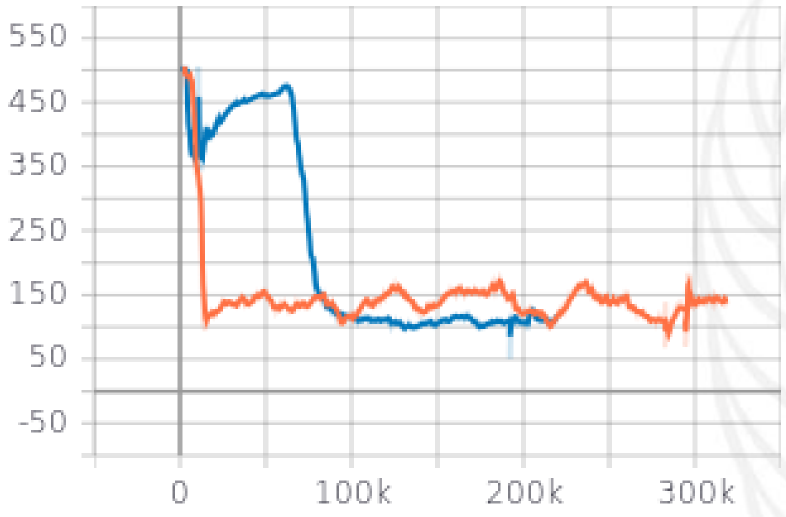


Min Rewards **Mean Rewards** **Max Rewards**

Results - Episodes

γ = 0.7 in orange and γ = 0.9 in blue

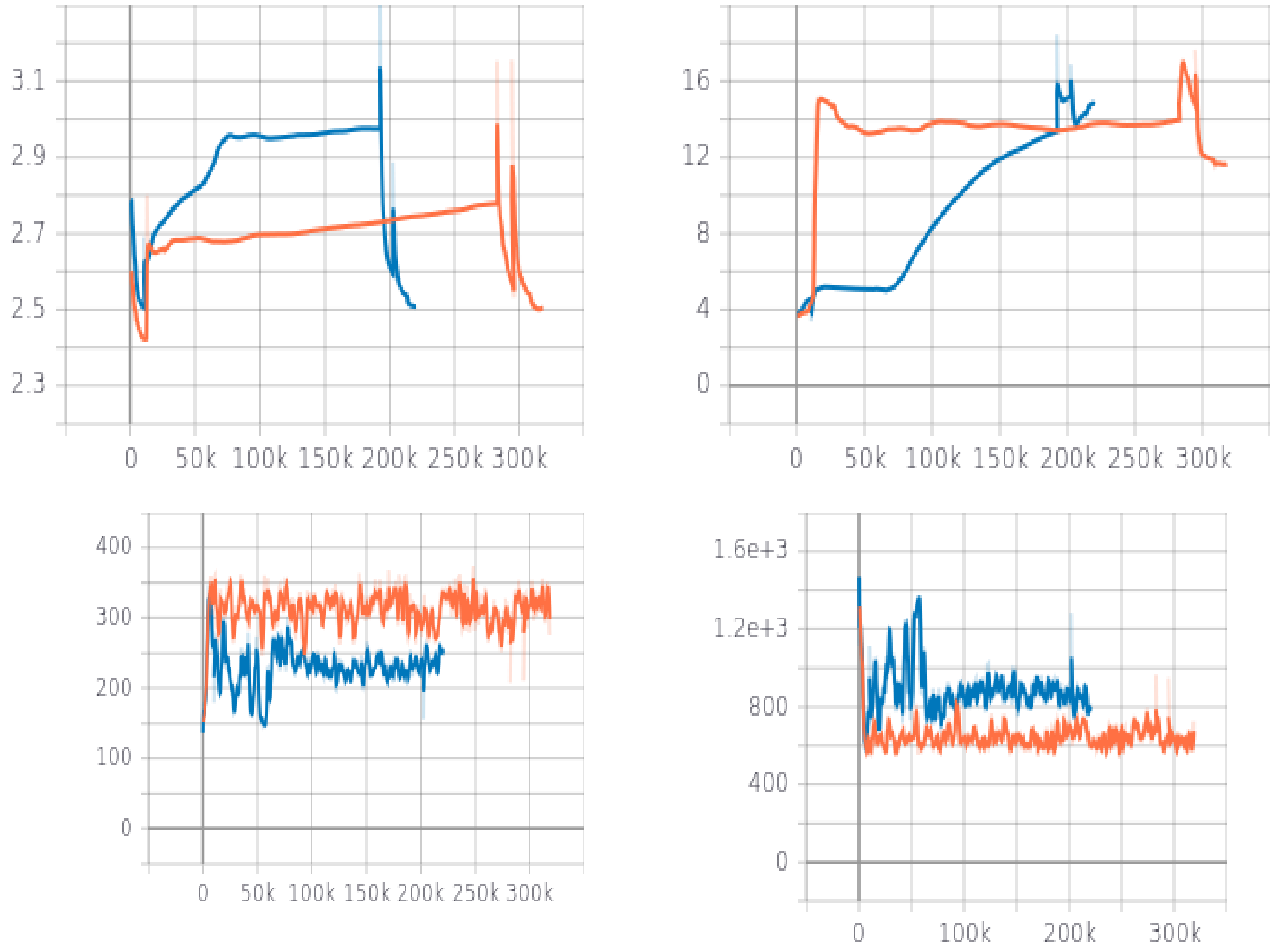
The mean length for episodes is similar for both γ values



Mean Episode Length

Results - Performance

γ = 0.7 in orange and γ = 0.9 in blue

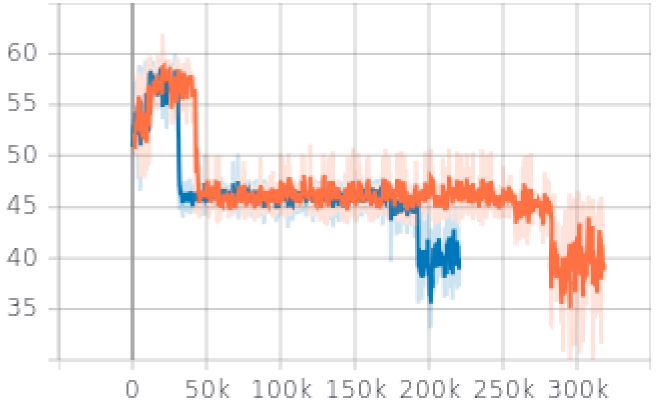


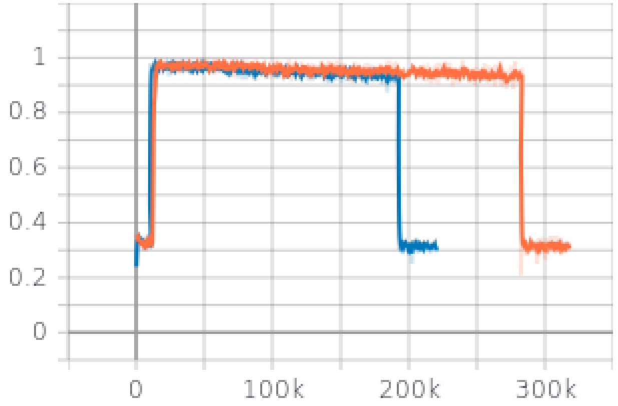
Performance-wise, γ = 0.7 gives higher throughput and takes less time than γ = 0.9

A. Inference time (ms)
B. Processing time (ms)
C. Learning throughput
D. Learning time

Results - Resources

γ = 0.7 in orange and γ = 0.9 in blue





As seen from all graphs, γ = 0.7 was also faster and so completed more iterations than γ = 0.9 in the same time

Resource utilization is same for both.

This is as expected because most of the resources are used for the CARLA simulation and the neural net.

γ value doesn’t have any significant impact on it.

Results - Example

Thank you!

Salil Dabholkar
50321748

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences