AWS DeepRacer

Team JCAR:

Jaydyn Odor, Cristopher Torres,
Ahmed Moustafa, and Ramiro Gonzalez

NACME
NATIONAL ACTION COUNCIL FOR MINORITIES IN ENGINEERING

UNIVERSITY OF ARKANSAS

Google

Tech Stack

What is AWS DeepRacer?

AWS DeepRacer is a global racing competition where users train a 1/18th scale car to race autonomously using **Reinforcement Learning**.

There is both a virtual and physical aspect to the competition:

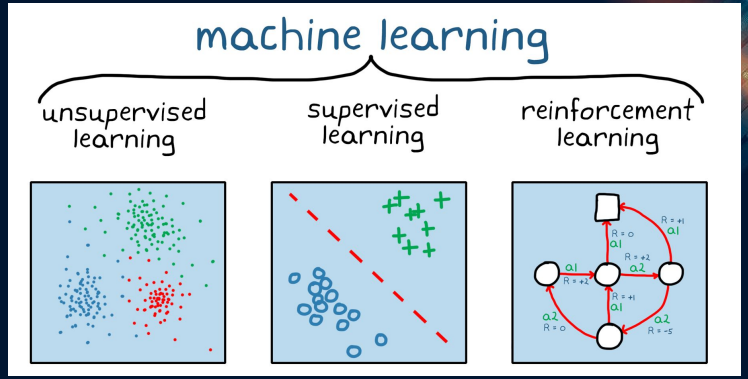
- Training as well as virtual evaluation is done online through Amazon Web Services (**AWS**)
- There is a physical robot that takes the trained model and autonomously drives through a real-life track

Intro to Reinforcement Learning

Formal Definition: *an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of* ***cumulative reward****.*

4 Elements of Reinforcement Learning:

1. **Agent**: program that is being trained
2. **Environment**: world in which agent performs actions
3. **Action**: 'move' made by agent which changes environment's state
4. **Rewards**: assigned evaluation of an action (+/-)

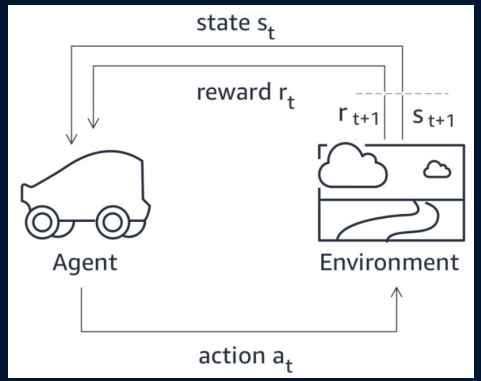


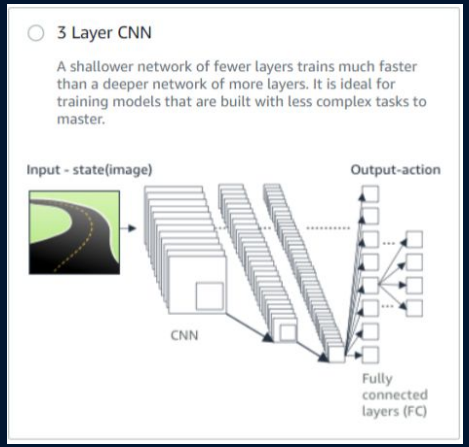
Unlike with Supervised Learning, reinforcement learning starts with random actions so the agent learns a variety of environment conditions. **This makes the trained model robust.**

How does DeepRacer work?

- The AWS DeepRacer service initializes the simulation with a virtual **track + background** as the environment and a **vehicle** as the agent.

- The agent embodies a **policy neural network** that can be tuned with hyper-parameters. The agent acts (as specified with a steering angle and a speed) based on a given **state** (represented by an image from the front camera).



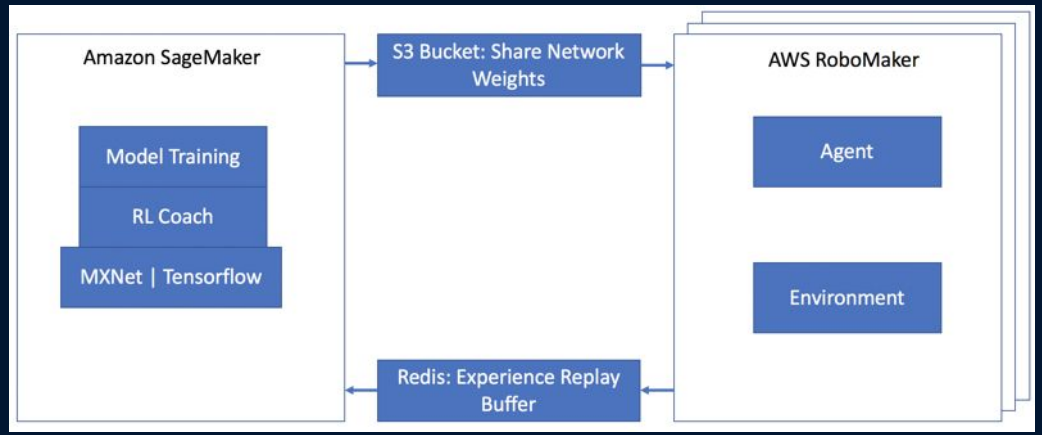


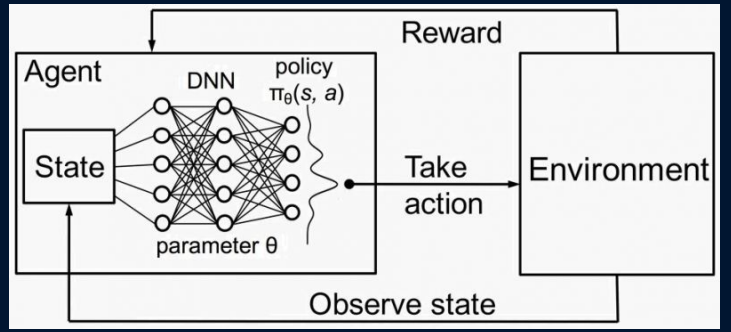
AWS DeepRacer default discrete action space

Action number	Steering	Speed
0	-30 degrees	0.4 m/s
1	-30 degrees	0.8 m/s
2	-15 degrees	0.4 m/s
3	-15 degrees	0.8 m/s
4	0 degrees	0.4 m/s
5	0 degrees	0.8 m/s
6	15 degrees	0.4 m/s
7	15 degrees	0.8 m/s
8	30 degrees	0.4 m/s
9	30 degrees	0.8 m/s

How does DeepRacer work? (cont.)

- The simulated environment updates the agent's position based on the selected action and **returns a reward** and an updated camera image.
- The old state, action, reward, and new state are used to update the neural network periodically. The updated network models are used to **create more experiences**.
- The AWS DeepRacer service periodically saves the neural network model to persistent storage (S3).





Reward Function Components

Progress Incentive

- Steps
- 15 steps/sec x 18 sec/lap = 270 steps per lap
- Progress
- Reward car for progressing faster than 18 sec/lap

```
# Read input parameters
distance_from_center = params['distance_from_center']
track_width = params['track_width']
speed = params['speed']
waypoints = params['waypoints']
closest_waypoints = params['closest_waypoints']
progress = params['progress']
steps = params['steps']

# Calculate 3 marks that are increasingly farther away from the center line
marker_1 = 0.1 * track_width
marker_2 = 0.25 * track_width
marker_3 = 0.5 * track_width

# initialize thresholds and reward
DIFF_HEADING_THRESHOLD = 6
SPEED_THRESHOLD = 1.8
TOTAL_STEPS = 270      # roughly 15 steps per second, 18 sec default lap
reward = 5
```

```
############################
# Steps and progress, check every 20 steps. The less steps it takes to reach 100% progress means finished faster
############################

# reward less steps with greater progress if faster than 18 sec
if (steps % 20) == 0 and progress/100 > (steps/TOTAL_STEPS):
    reward += progress - (steps/TOTAL_STEPS)*100
```

Waypoints Incentive

- Penalize fast driving along turns
- waypoints
- closest_waypoints
- Calculate angle car is currently facing
- Calculate angle to face future waypoint

```
##############################
# Waypoints: referenced code from https://github.com/MatthewSuntup/DeepRacer/blob/master/reward/reward_final.py
##############################

# finding previous point, next point, and future point
prev_point = waypoints[closest_waypoints[0]]
next_point = waypoints[closest_waypoints[1]]
future_point = waypoints[min(len(waypoints) - 1, closest_waypoints[1]+6)]

# calculate headings to waypoints
heading_current = math.degrees(math.atan2(prev_point[1]-next_point[1], prev_point[0]-next_point[0]))
heading_future = math.degrees(math.atan2(prev_point[1]-future_point[1], prev_point[0]-future_point[0]))

# calculate difference between headings
# check we didn't choose reflex angle
diff_heading = abs(heading_current-heading_future)
if diff_heading > 180:
    diff_heading = 360 - diff_heading

# if diff_heading > than threshold indicates turn
# so when a turn is ahead (high diff_heading)
# penalize high speed, reward slow speed
if (diff_heading > DIFF_HEADING_THRESHOLD) and speed >= SPEED_THRESHOLD:
    reward -= 4
```

Centerline Incentive

```
############################
# Center line incentives
############################

# give higher reward if the car is closer to center line and vice versa
if distance_from_center <= marker_1:
    reward += 5
elif distance_from_center <= marker_2:
    reward += 4
elif distance_from_center <= marker_3:
    reward += 3
else:
    reward -= 4
```

Virtual Training

Training configuration

Environment simulation
re:Invent 2018

Reward function
Show

Sensor(s)
Camera

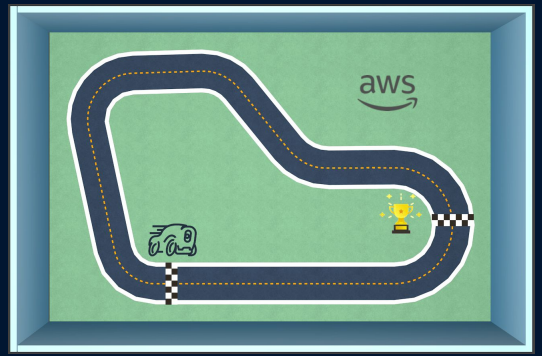
Action space type
Continuous

Action space
Speed: [0.6 : 3] m/s
Steering angle: [-30 : 30] °

Framework
Tensorflow

Reinforcement learning algorithm
PPO

Hyperparameter	Value
Gradient descent batch size	64
Entropy	0.01
Discount factor	0.999
Loss type	Huber
Learning rate	0.0003
Number of experience episodes between each policy-updating iteration	20
Number of epochs	10

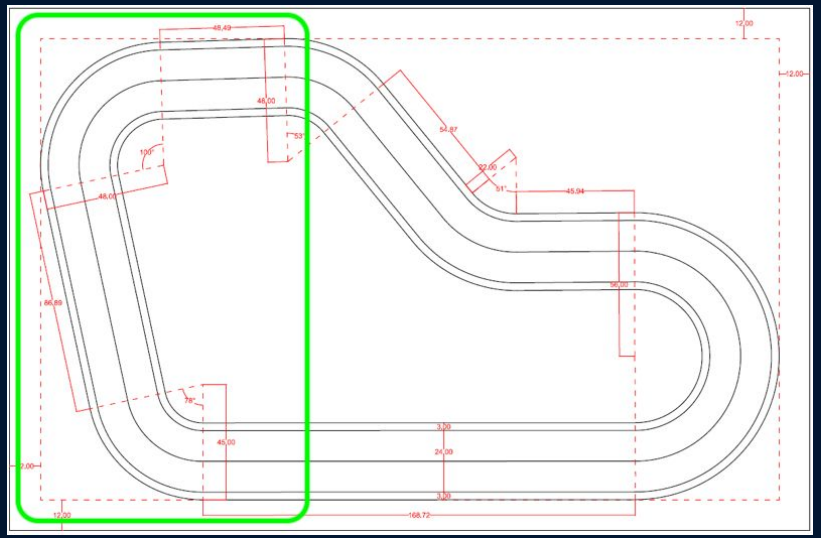


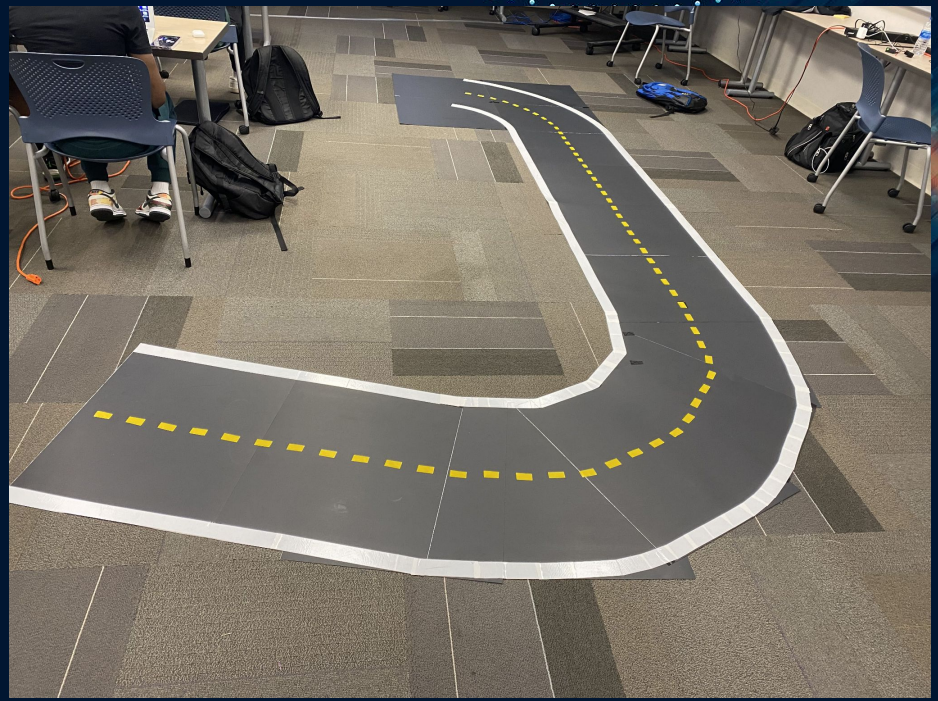
Virtual Evaluation

Trial	Time (MM:SS.mmm)
1	00:13.191
2	00:13.138
3	00:13.471

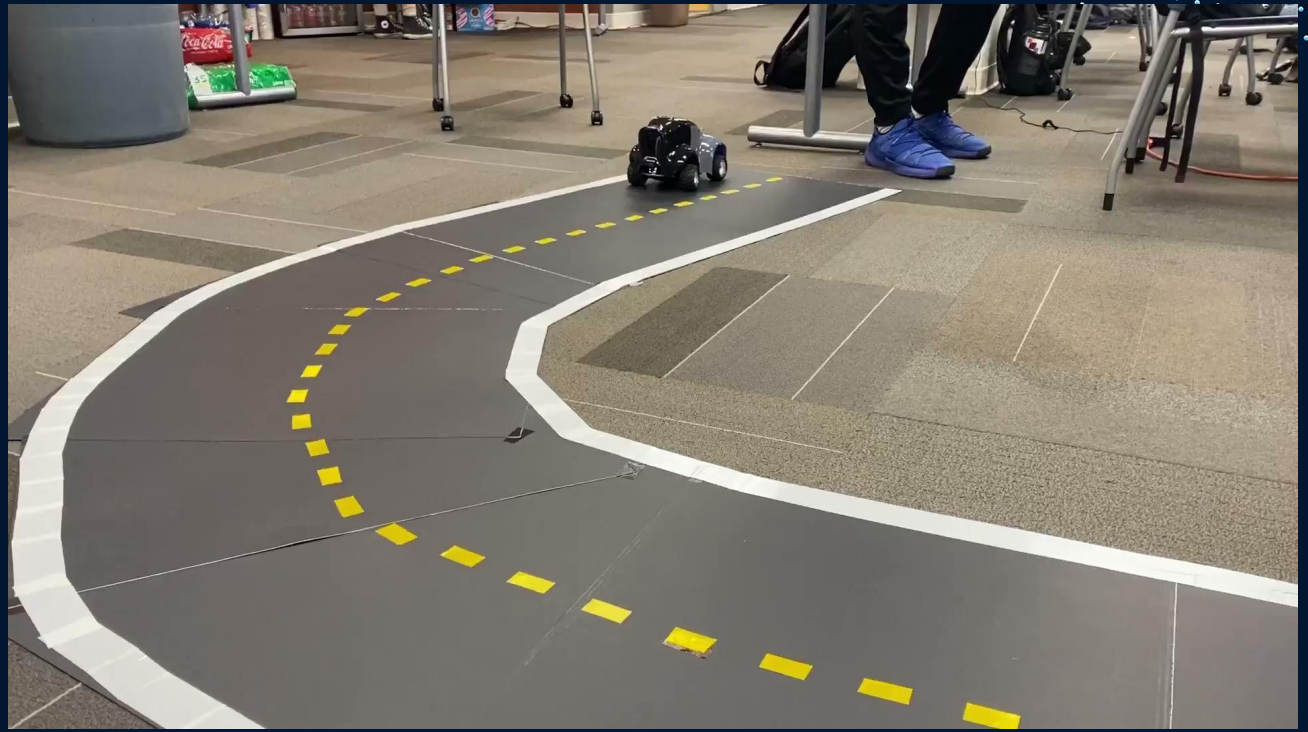
Physical Car Assembly

Physical Track





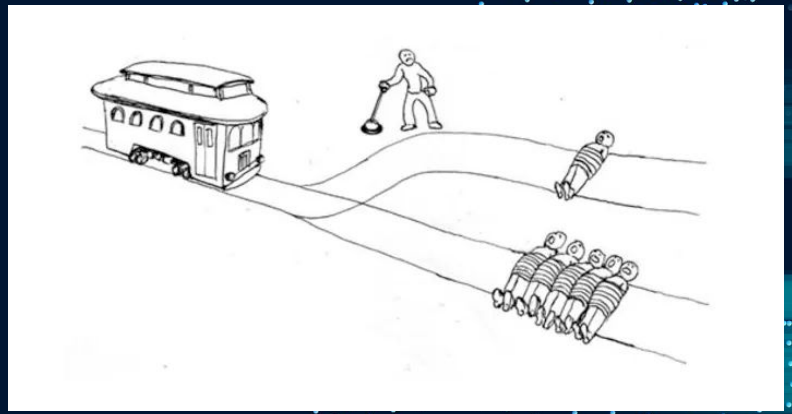
Physical Evaluation



Ethical Considerations

- Ensure fairness among users
- Adaptability for different settings
- Reliability and safety precautions
- Inability of moral

- Road conditions / weather bias
- Road obstruction classification
- Civilian recognition bias
- Negative impact on labor

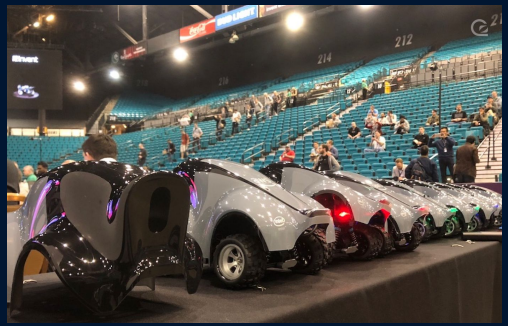


Limitations / Issues

- **WiFi limitations** made connecting our laptop to the robot and streaming the camera's content sluggish and sometimes non-existent. University WiFi was preventing communication between the car and our computers. Our solution was to bring our own router to create a separate network that both the laptop and car could connect to.
- The **AWS DeepRacer Compute Module** caused many issues that were nearly undiagnosable. The webserver for controlling the car would not open on our computers when connected to the car even after factory resetting by installing a fresh Ubuntu OS onto it and resetting the AWS credentials. The solution was to add an "http" rule to the compute module.
- Our free AWS accounts only had **10 hours for training & testing models** which made it hard to play around with all the combinations of hyperparameters as well as the fine-tuning for the reward function values which may have improved our time. Our approach to mitigate this was having each person test different models on separate accounts to get the most efficiency.

Future Goals

- Optimization of our model to be faster than 10 seconds per lap
- Testing more on other tracks to avoid overfitting to one track
- Build a more permanent track using EVA foam pieces so that it is collapsable and transportable.
- Both virtually and physically, entering a competition is something that would allow us to compare and discuss our models and training environments to others who may have more experience or understanding in the field of Reinforcement Learning and Autonomous Driving.



Special Thanks to

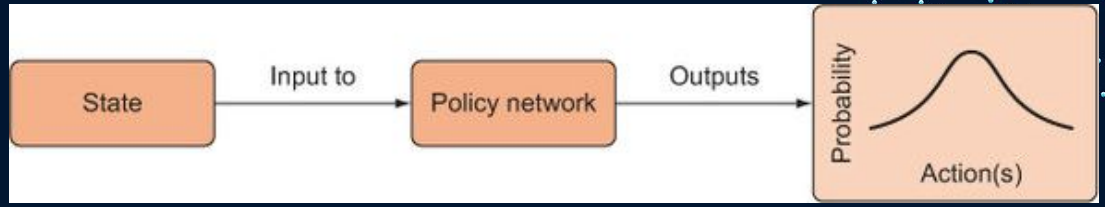
- NACME, Google, and Uark
- Michele Lezama
- Dr. Chase Rainwater
- Dr. Bryan Hill and Teresa Simpson
- Jordy Morquecho
- Luc Gabbard (Uark IT)

Resources

- https://www.baeldung.com/cs/mdp-value-iteration
- https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292
- https://docs.aws.amazon.com/deepracer/latest/developerguide/deepracer-how-it-works-action-space.html
- https://docs.aws.amazon.com/deepracer/latest/developerguide/deepracer-how-it-works-solution-workflow.html
- https://github.com/MatthewSuntup/DeepRacer/blob/master/reward/reward_final.py
- https://towardsdatascience.com/an-advanced-guide-to-aws-deepracer-2b462c37eea

Questions?

Policy Iteration (if time permits)



choose an arbitrary policy π';

loop

$\pi = \pi'$;

compute the value function of policy π (**policy evaluation**):

solve the linear equations

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^{\pi}(s');$$

improve the policy at each state (**policy improvement**):

$$\pi'(s) = \arg\max_{a} \left\{ R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^{\pi}(s') \right\};$$

until $\pi = \pi'$;