AWS DeepRacer: Get hands-on with machine learning

Agenda

- Introducing AWS DeepRacer
- Introduction to machine learning
- Reinforcement learning key terms
- The reward function
- AWS DeepRacer ML architecture
- Demo

Introducing AWS DeepRacer

How can we put machine learning in the hands of all developers?

Literally

A closer look

- 1:18 4WD scale car
- Intel Atom processor
- Intel distribution of OpenVINO toolkit
- Front-facing camera (4 megapixels)
- System memory: 4 GB RAM
- 802.11ac Wi-Fi
- Ubuntu 20.04 Focal Fossa
- ROS 2 Foxy Fitzroy

AWS DeepRacer Evo Expansion Pack

- Second front-facing camera (stereo cameras)
- 360-degree, 12-meter scanning radius Lidar sensor

OpenVINO™

intel. ATOM

Get racing…

Get racing…

3D racing
simulator

Get racing…

3D racing simulator

AWS DeepRacer League

Get racing…

3D racing simulator

AWS DeepRacer League

Community races

Get racing…

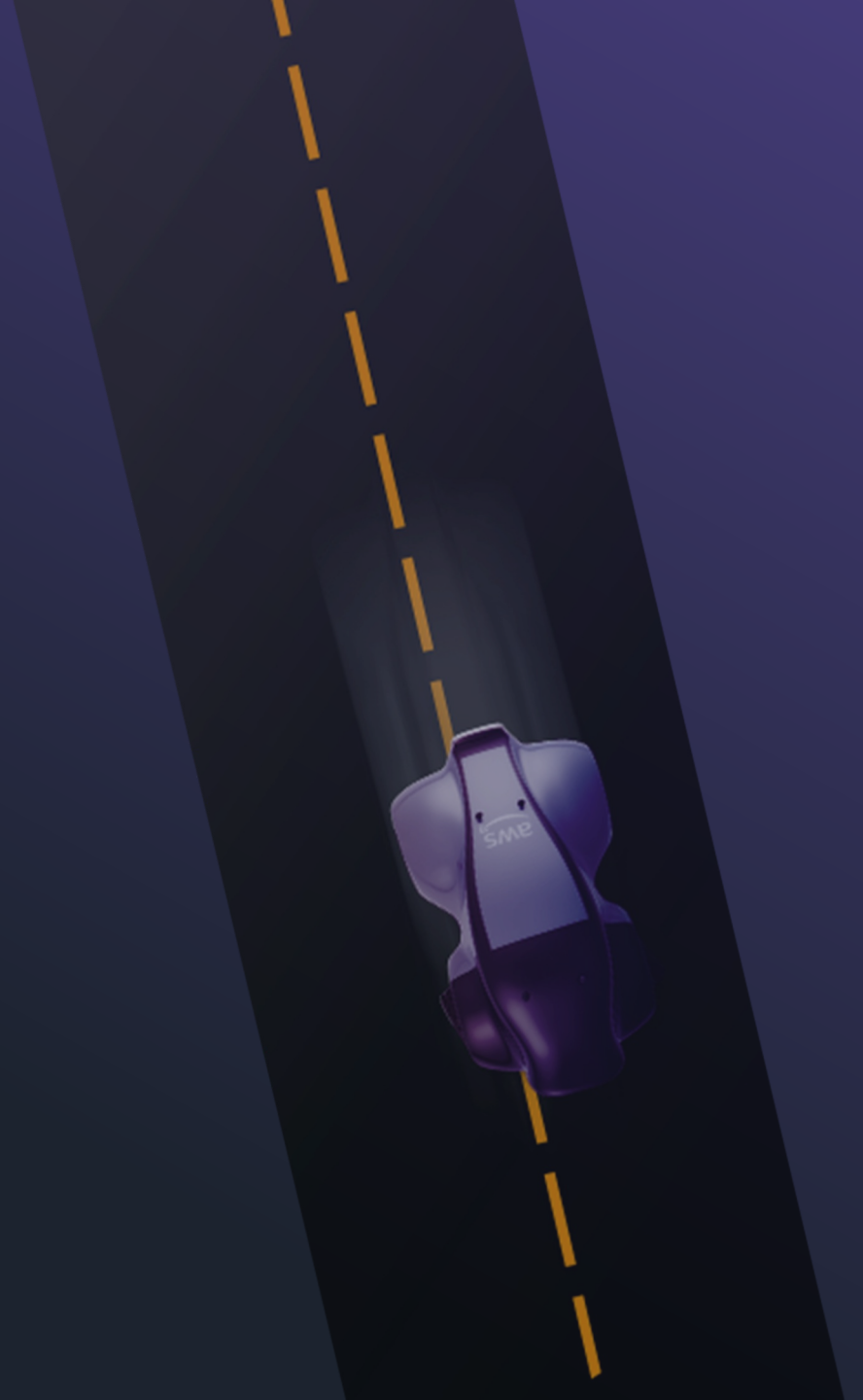
3D racing simulator

AWS DeepRacer League

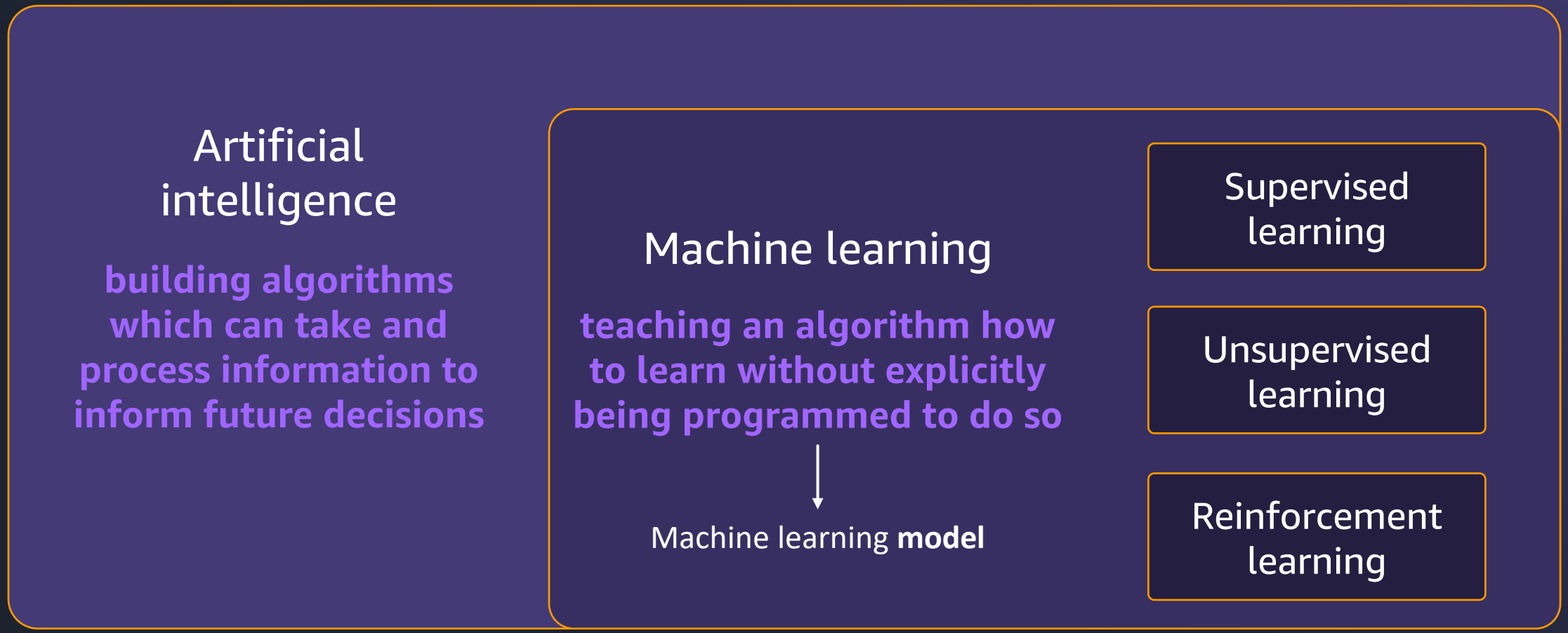
Community races

AWS DeepRacer

Introduction to machine learning



The broader context



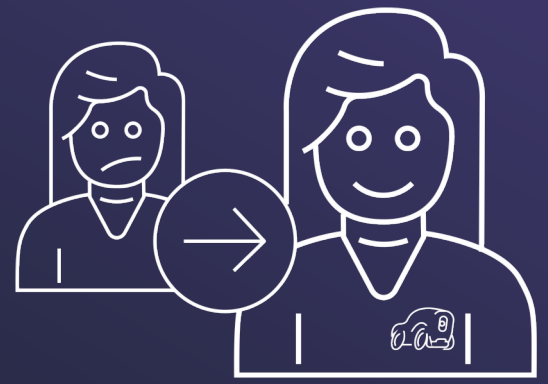
ML overview

Supervised

Example-driven training; every piece of data has a corresponding label

Unsupervised

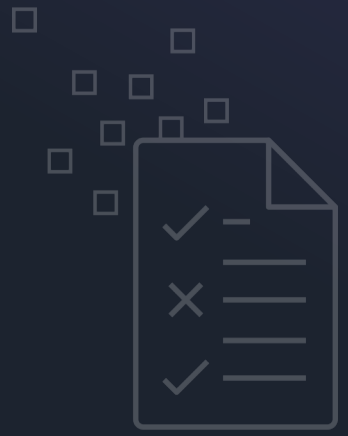
No labels for training data; useful for clustering like data, discovering patterns for generative AI



Reinforcement

Learns through consequences of actions in a specific environment

ML overview

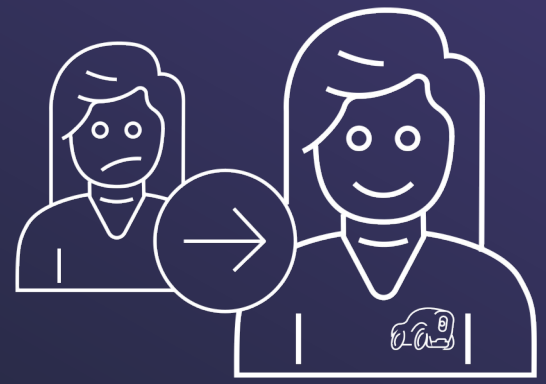


Supervised

Example-driven training; every piece of data has a corresponding label

Unsupervised

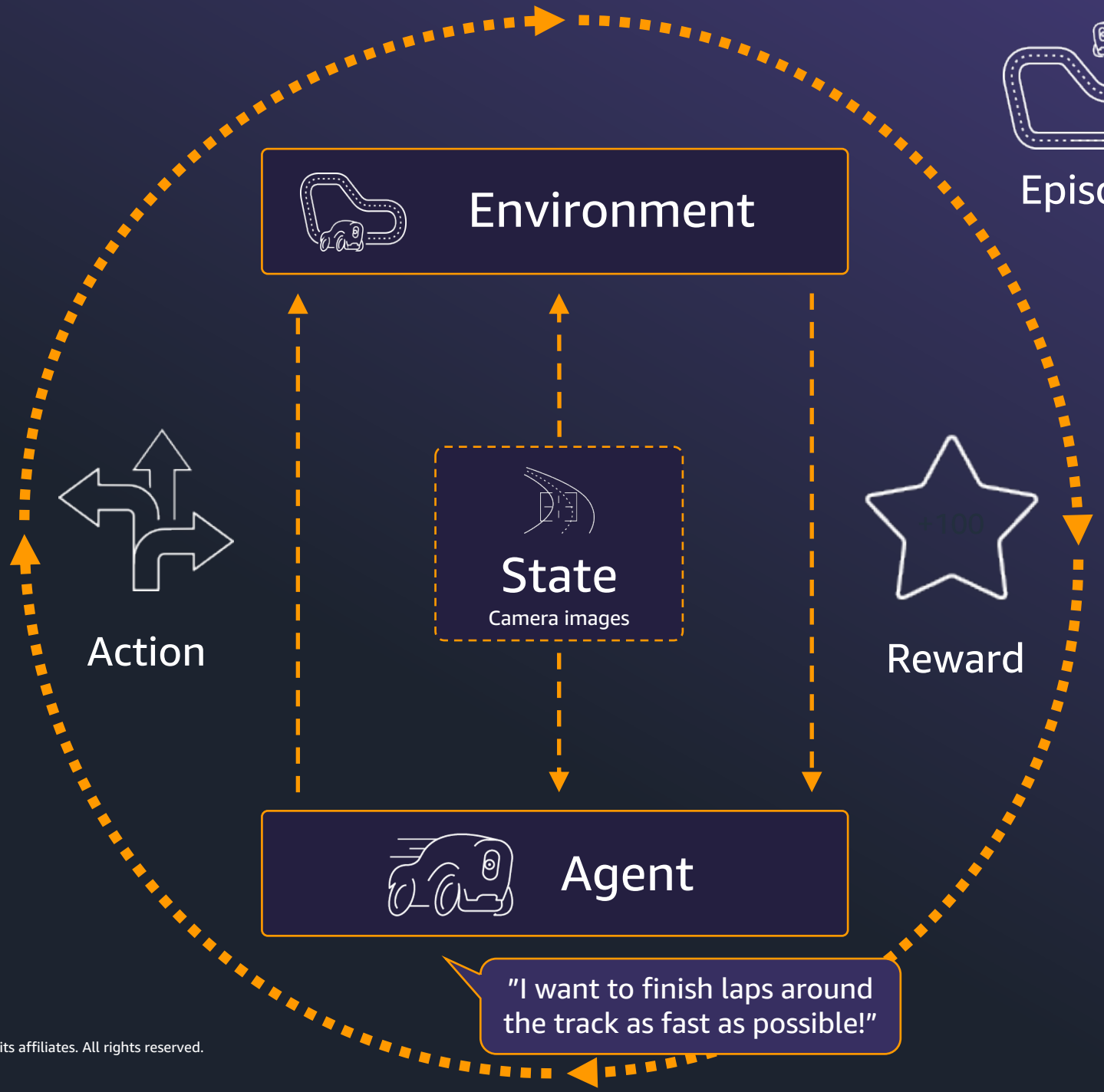
No labels for training data; useful for clustering like data, discovering patterns for generative AI



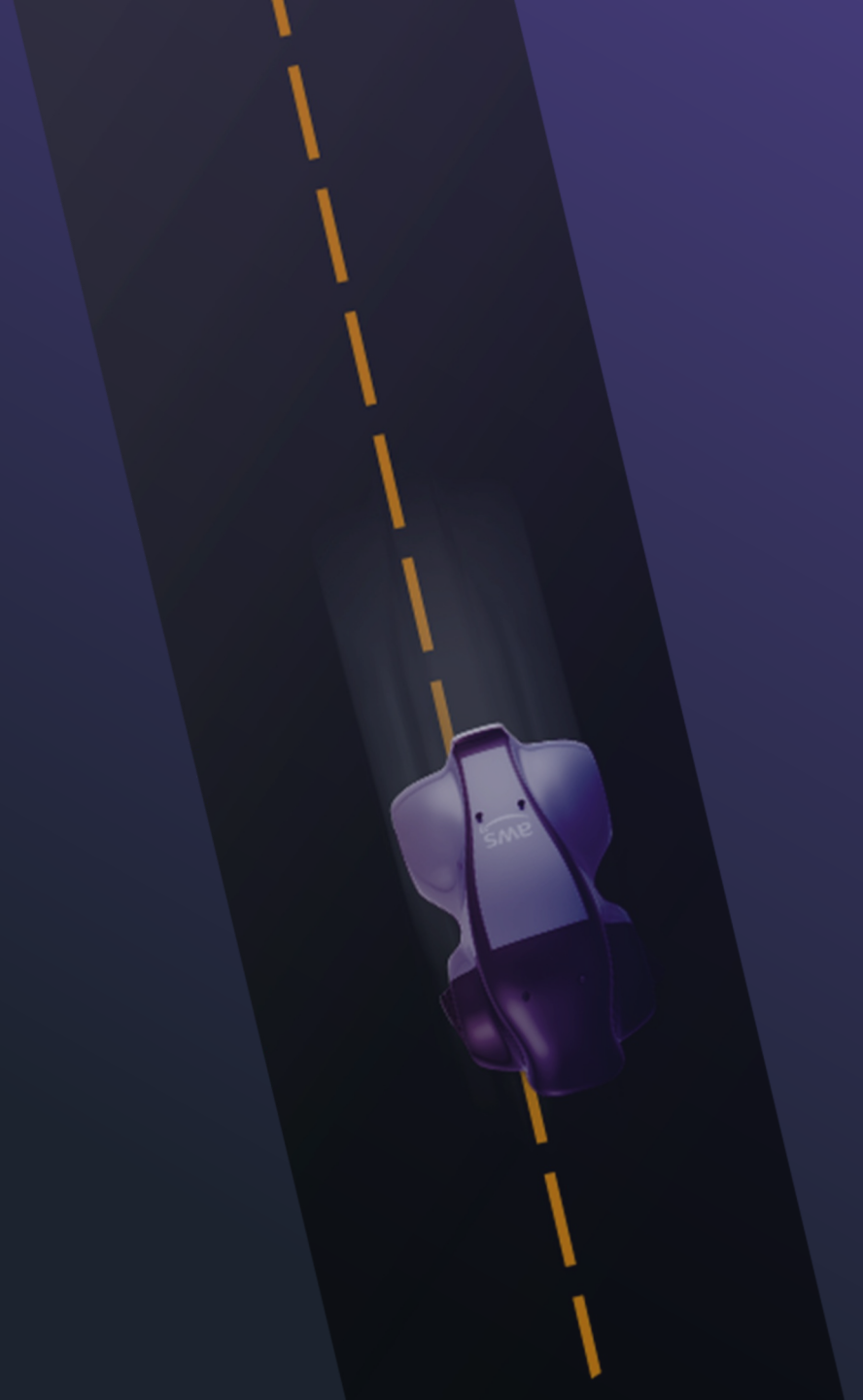
Reinforcement

Learns through consequences of actions in a specific environment

Reinforcement learning key terms



The reward function

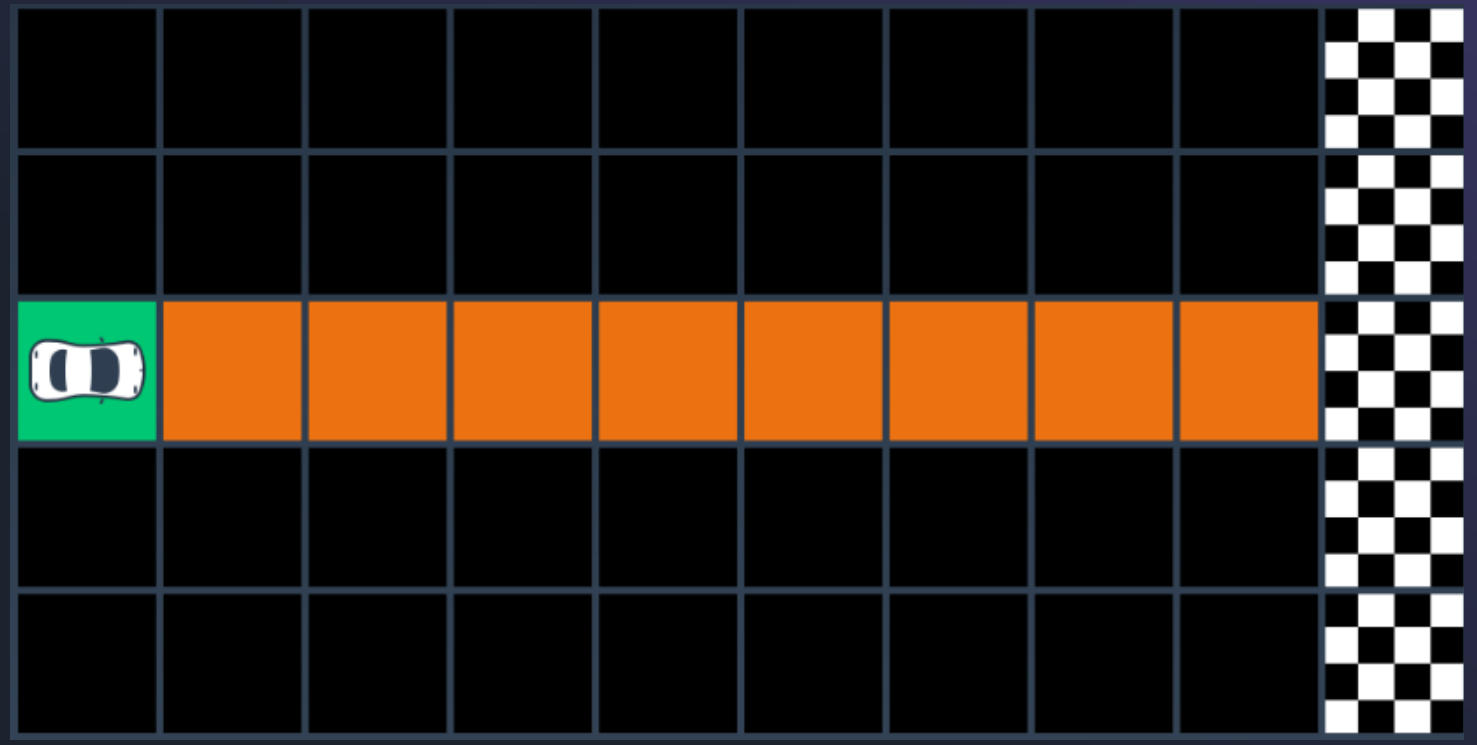


The reward function

The reward function incentivizes particular behaviors and is at the core of reinforcement learning

The reward function: Straight track race

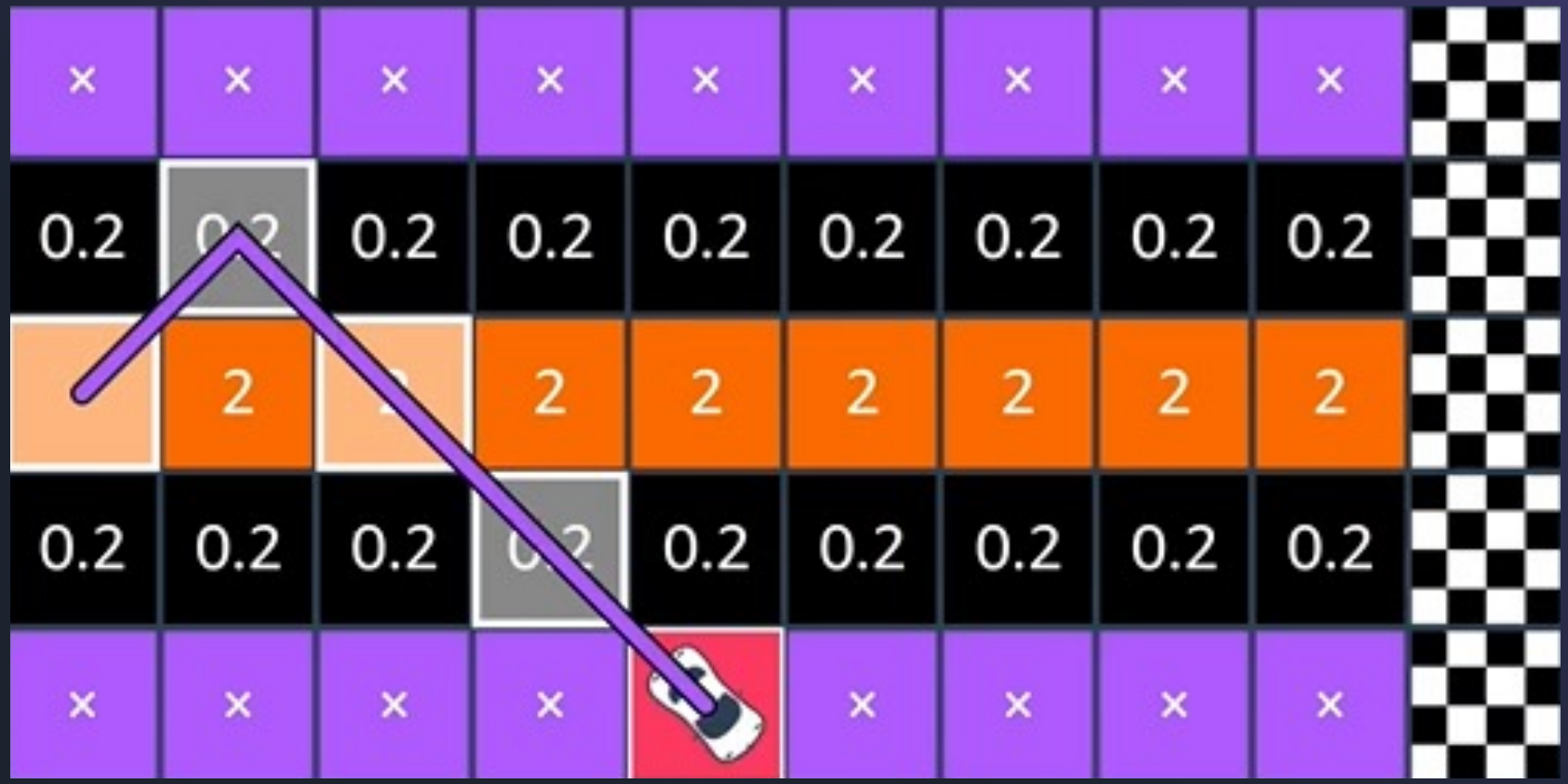
Agent



Goal

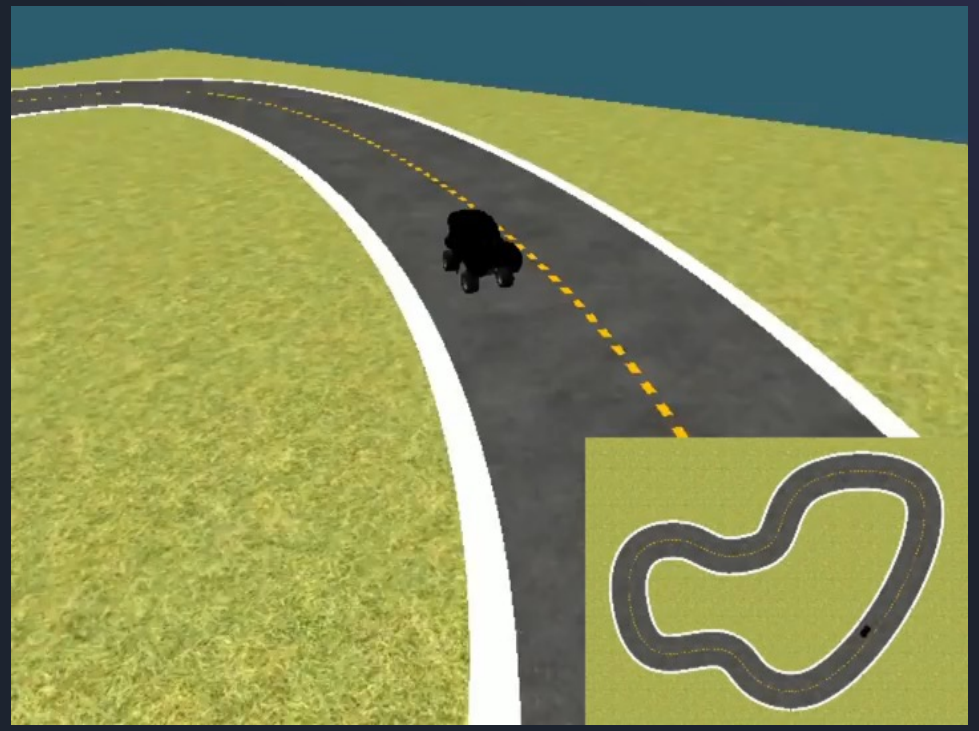
Rewards that incentivize center-line driving

Exploration versus exploitation

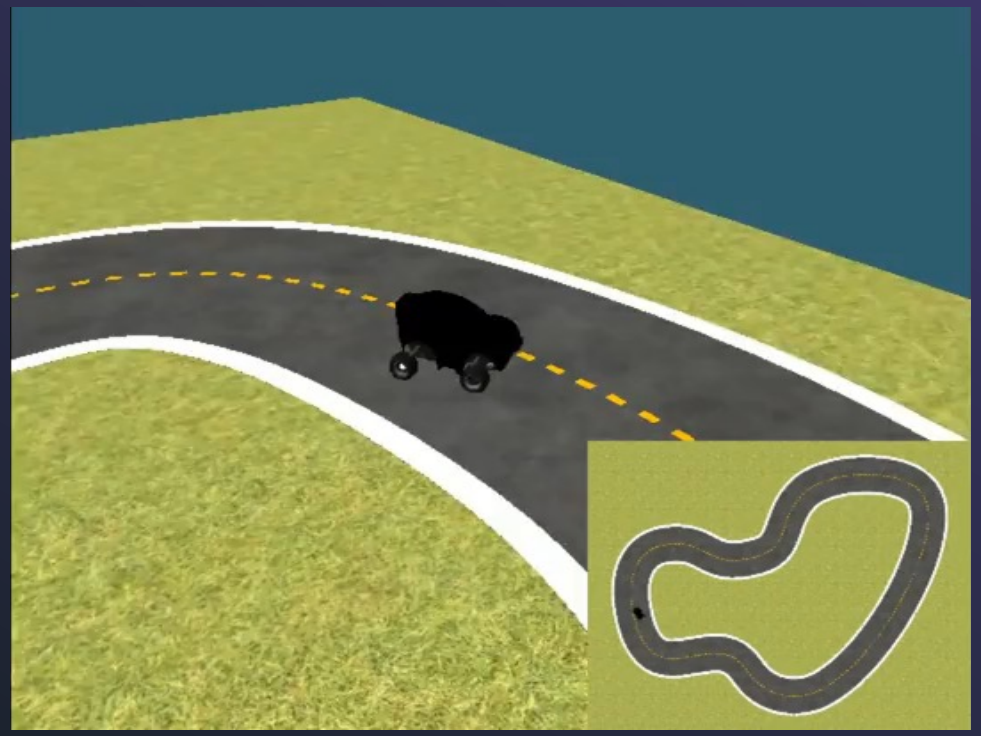


Exploration versus exploitation

Exploration

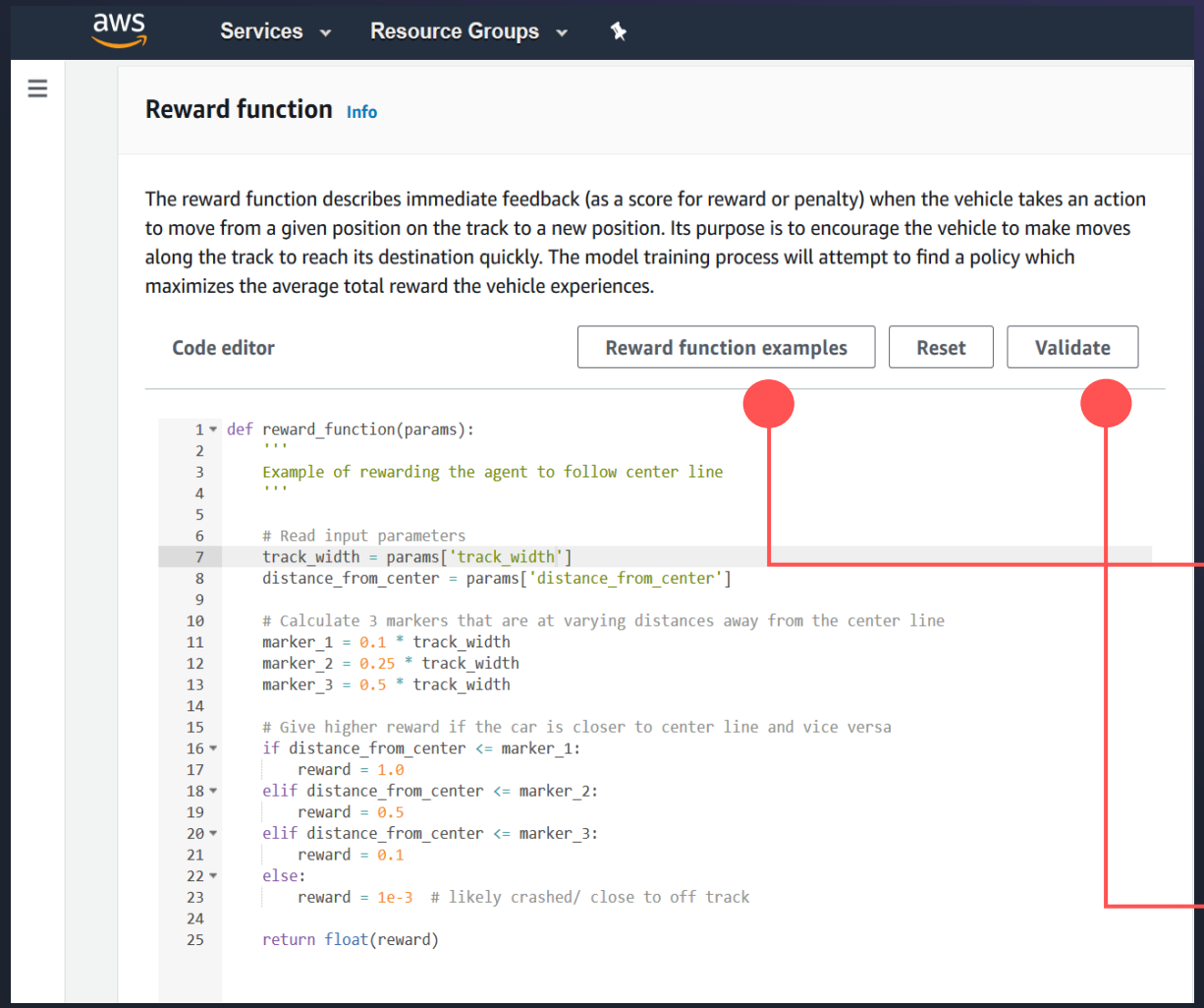


Exploitation



Convergence

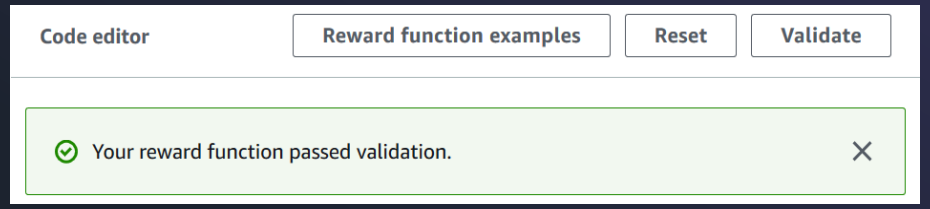
Programming your own reward function



Code editor: Python 3 syntax

Three example reward functions

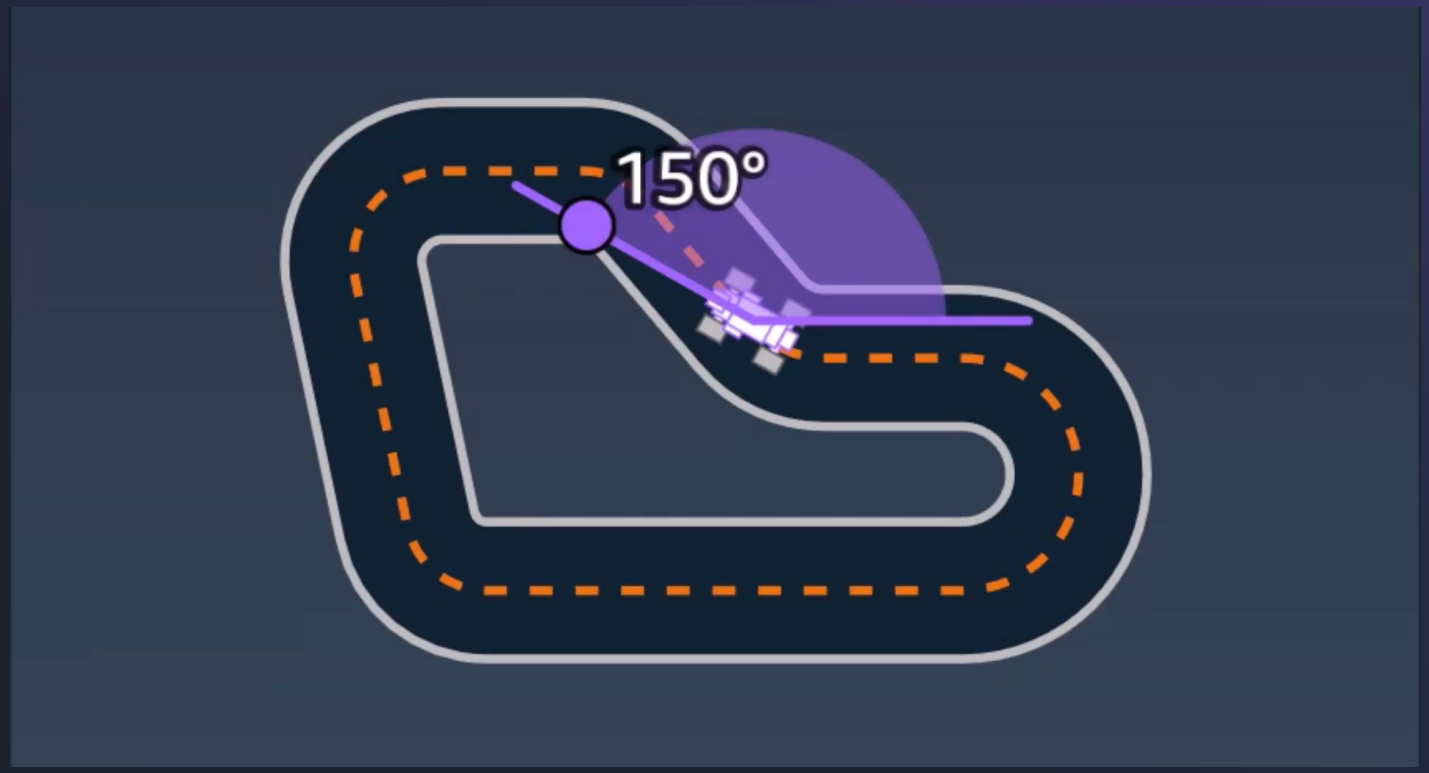
Code validation via Lambda



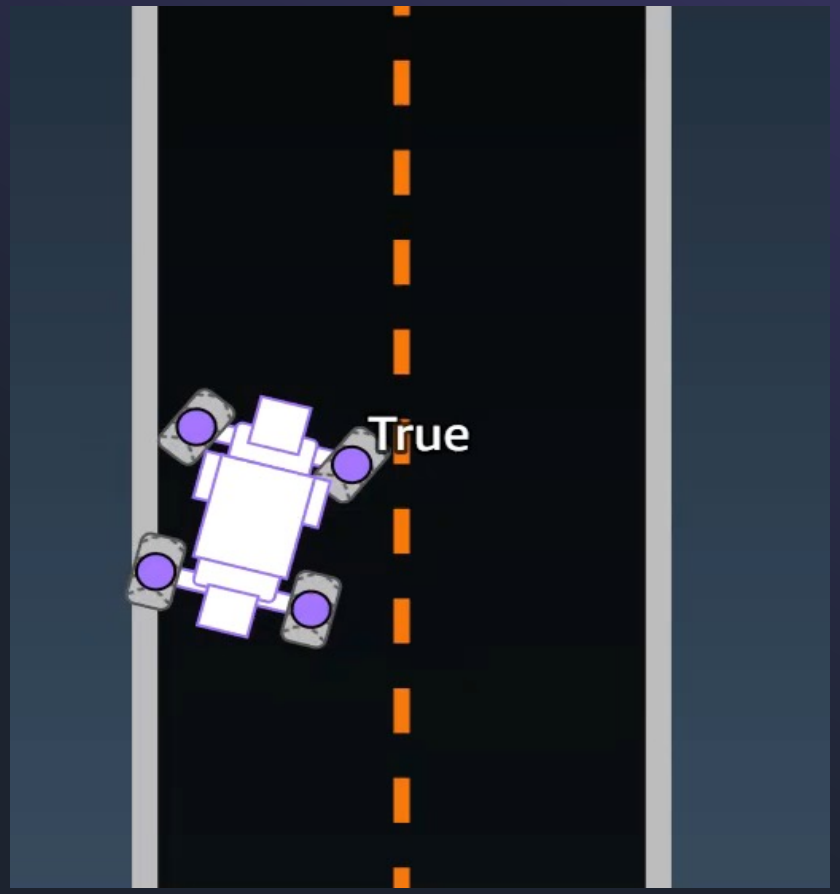
Reward function input parameters

```
{
    "all_wheels_on_track": Boolean,        # flag to indicate if the agent is on the track
    "x": float,                            # agent's x-coordinate in meters
    "y": float,                            # agent's y-coordinate in meters
    "closest_objects": [int, int],         # zero-based indices of the two closest objects to the agent's current position of (x, y).
    "closest_waypoints": [int, int],       # indices of the two nearest waypoints.
    "distance_from_center": float,         # distance in meters from the track center
    "is_crashed": Boolean,                 # Boolean flag to indicate whether the agent has crashed.
    "is_left_of_center": Boolean,          # Flag to indicate if the agent is on the left side to the track center or not.
    "is_offtrack": Boolean,                # Boolean flag to indicate whether the agent has gone off track.
    "is_reversed": Boolean,                # flag to indicate if the agent is driving clockwise (True) or counter clockwise (False).
    "heading": float,                      # agent's yaw in degrees
    "objects_distance": [float, ],         # list of the objects' distances in meters between 0 and track_length in relation to the starting line.
    "objects_heading": [float, ],          # list of the objects' headings in degrees between -180 and 180.
    "objects_left_of_center": [Boolean, ], # list of Boolean flags indicating whether elements' objects are left of the center (True) or not (False).
    "objects_location": [(float, float),], # list of object locations [(x,y), ...].
    "objects_speed": [float, ],            # list of the objects' speeds in meters per second.
    "progress": float,                     # percentage of track completed
    "speed": float,                        # agent's speed in meters per second (m/s)
    "steering_angle": float,               # agent's steering angle in degrees
    "steps": int,                          # number steps completed
    "track_length": float,                 # track length in meters.
    "track_width": float,                  # width of the track
    "waypoints": [(float, float), ]        # list of (x,y) as milestones along the track center

}
```

Example parameter: heading



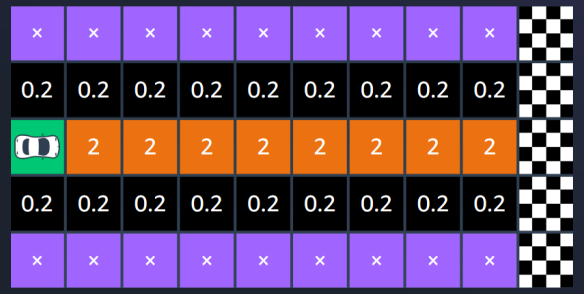
Example parameter: all_wheels_on_track



Example parameter: distance_from_center

Rewards that incentivize center-line driving

Rewards that incentivize center-line driving

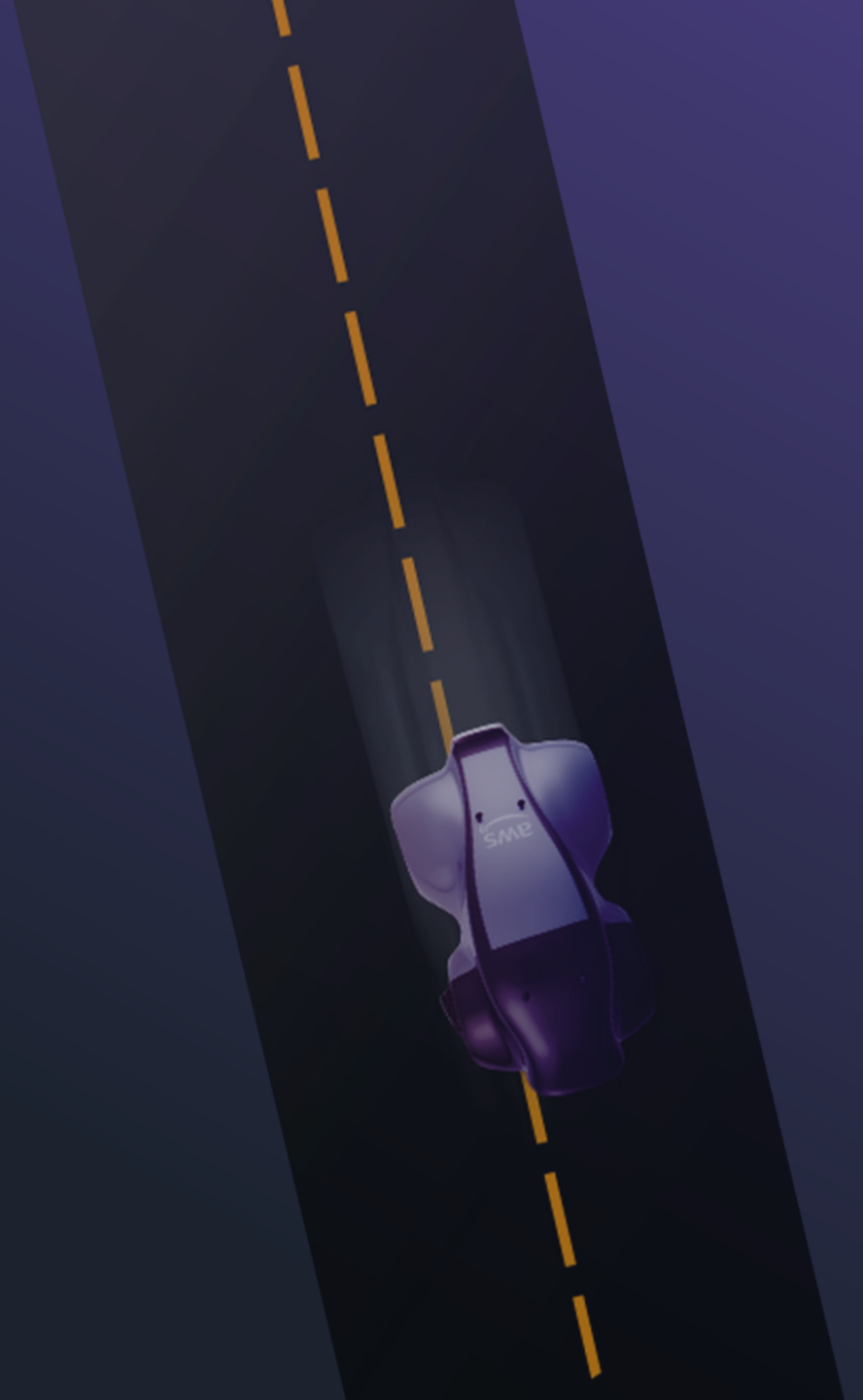



```
def reward_function(params):

    track_width = params['track_width']
    distance_from_center = params['distance_from_center']

    marker_1 = 0.1 * track_width
    marker_2 = 0.5 * track_width

    if distance_from_center <= marker_1:
        reward = 2.0 # Close to center line
    elif distance_from_center <= marker_2:
        reward = 0.2 # Further from center line
    else:
        reward = 1e-3 # Off track

    return float(reward)
```

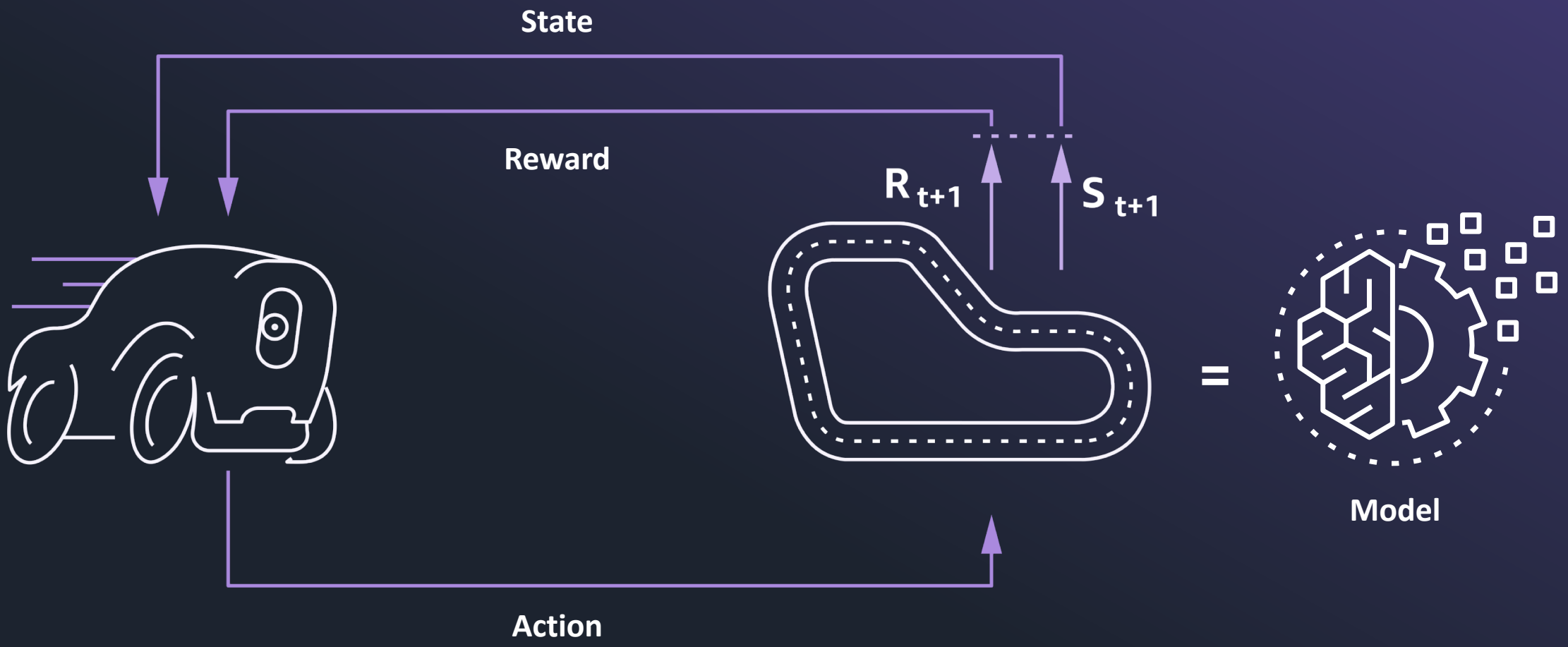
Sample reward functions

```python
def reward_function(params):
    '''
    Example of rewarding the agent to stay inside the two borders of the track
    '''

    # Read input parameters
    all_wheels_on_track = params['all_wheels_on_track']
    distance_from_center = params['distance_from_center']
    track_width = params['track_width']

    # Give a very low reward by default
    reward = 1e-3

    # Give a high reward if no wheels go off the track and
    # the agent is somewhere in between the track borders
    if all_wheels_on_track and (0.5*track_width - distance_from_center) >= 0.05:
        reward = 1.0

    # Always return a float value
    return float(reward)
```

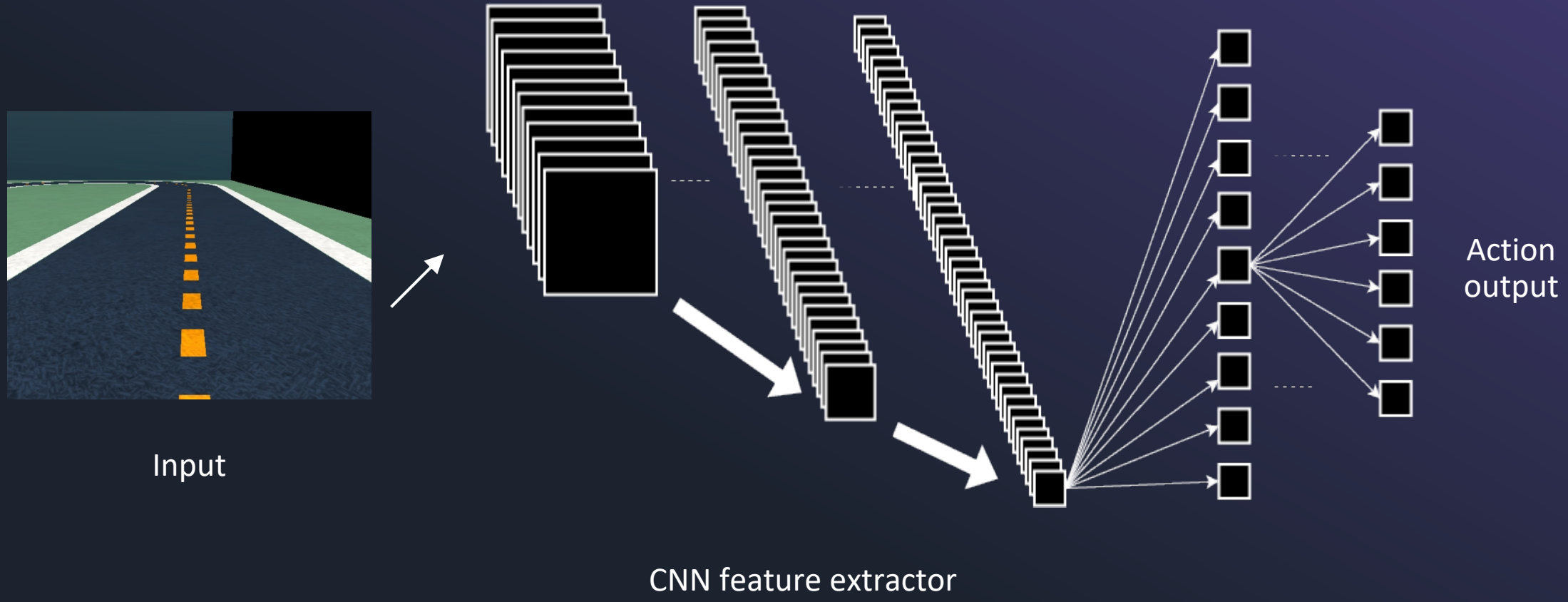
AWS DeepRacer ML architecture



How does the learning (training) happen?



AWS DeepRacer neural network architecture

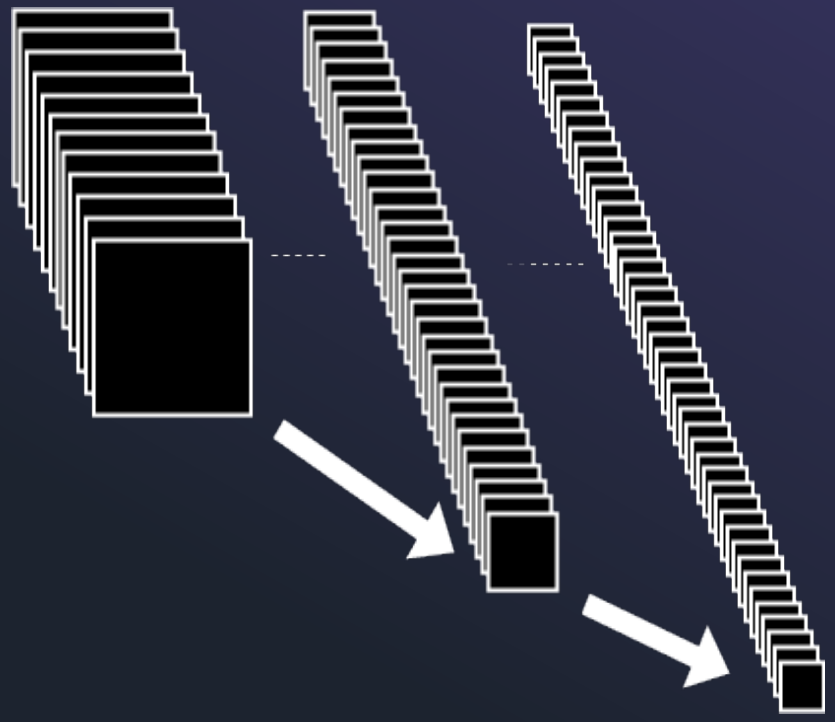


AWS DeepRacer neural network architecture

Input

AWS DeepRacer neural network architecture

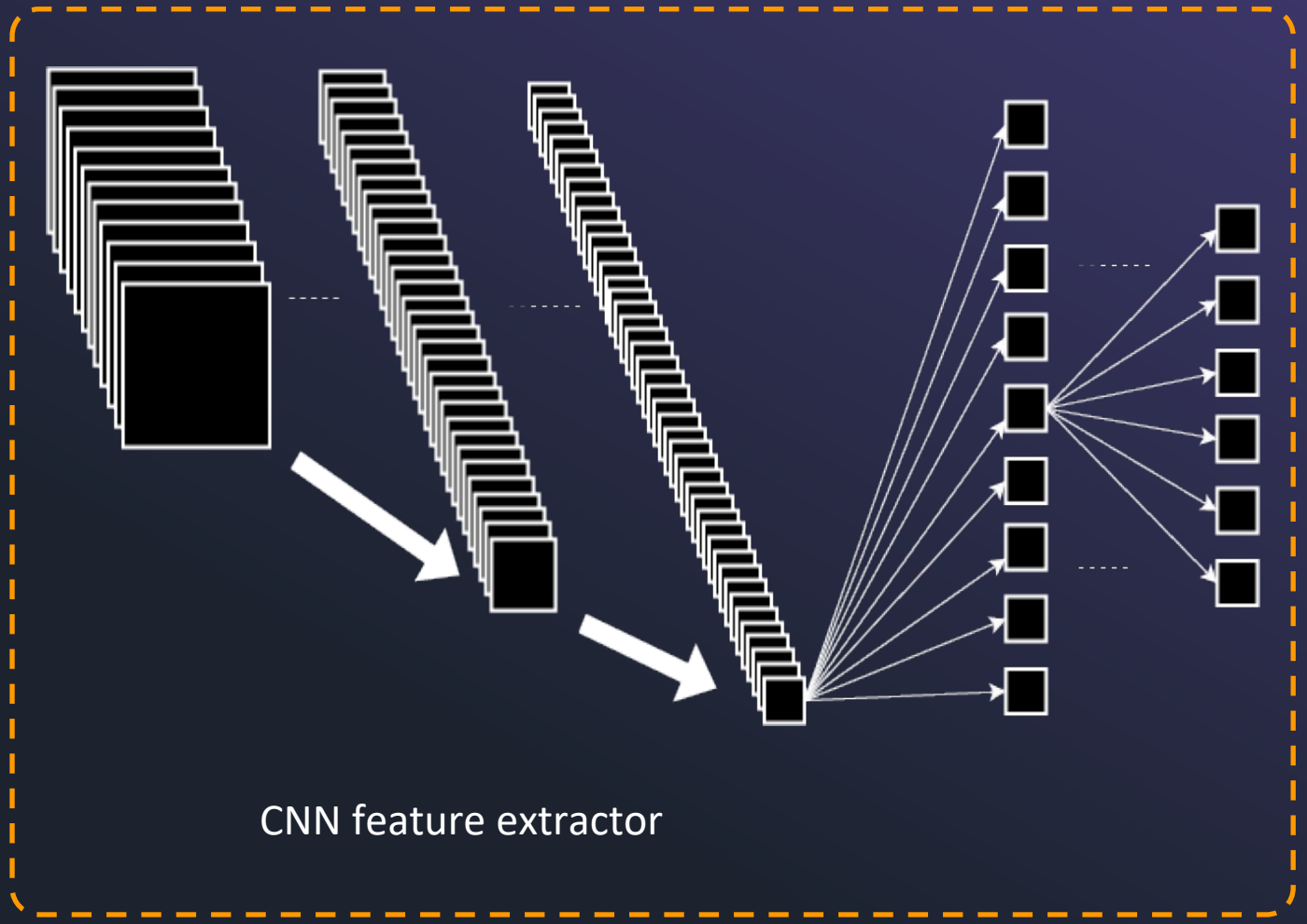
Input



CNN feature extractor

AWS DeepRacer neural network architecture

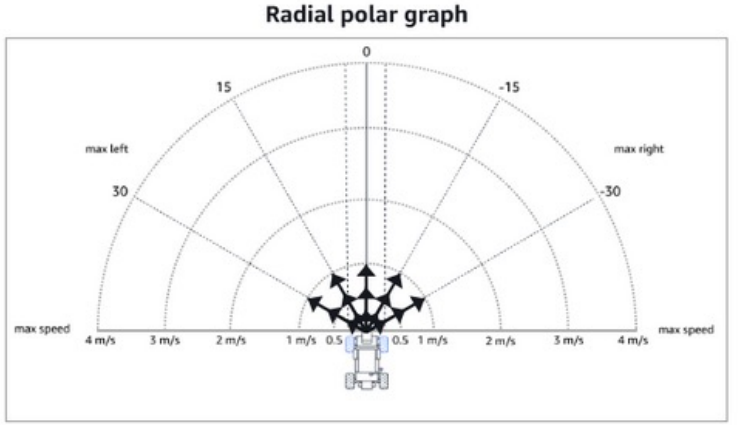
Input



Action space

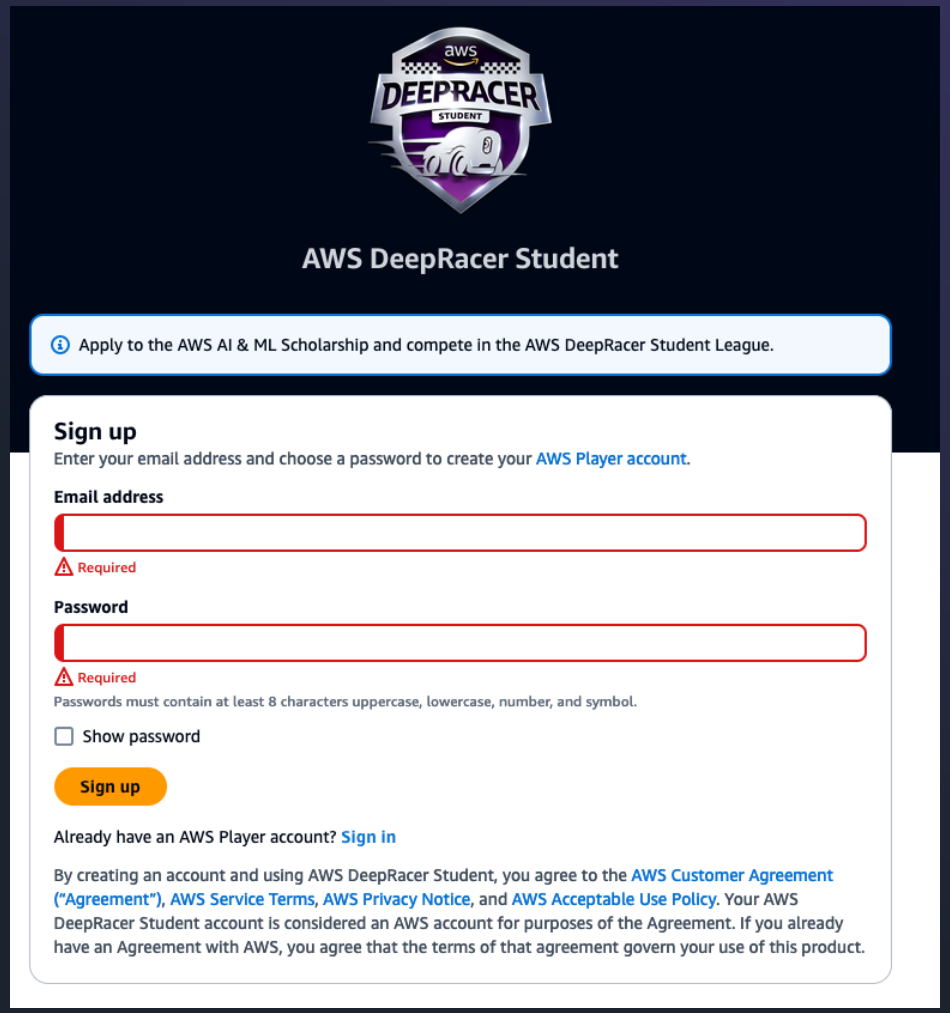
Action list — Advanced configuration

Action	Steering angle	Speed
0	-30.0 degrees	0.33 m/s
1	-30.0 degrees	0.67 m/s
2	-30.0 degrees	1.00 m/s
3	-15.0 degrees	0.33 m/s
4	-15.0 degrees	0.67 m/s
5	-15.0 degrees	1.00 m/s
6	0.0 degrees	0.33 m/s
7	0.0 degrees	0.67 m/s
8	0.0 degrees	1.00 m/s
9	15.0 degrees	0.33 m/s
10	15.0 degrees	0.67 m/s
11	15.0 degrees	1.00 m/s
12	30.0 degrees	0.33 m/s
13	30.0 degrees	0.67 m/s
14	30.0 degrees	1.00 m/s

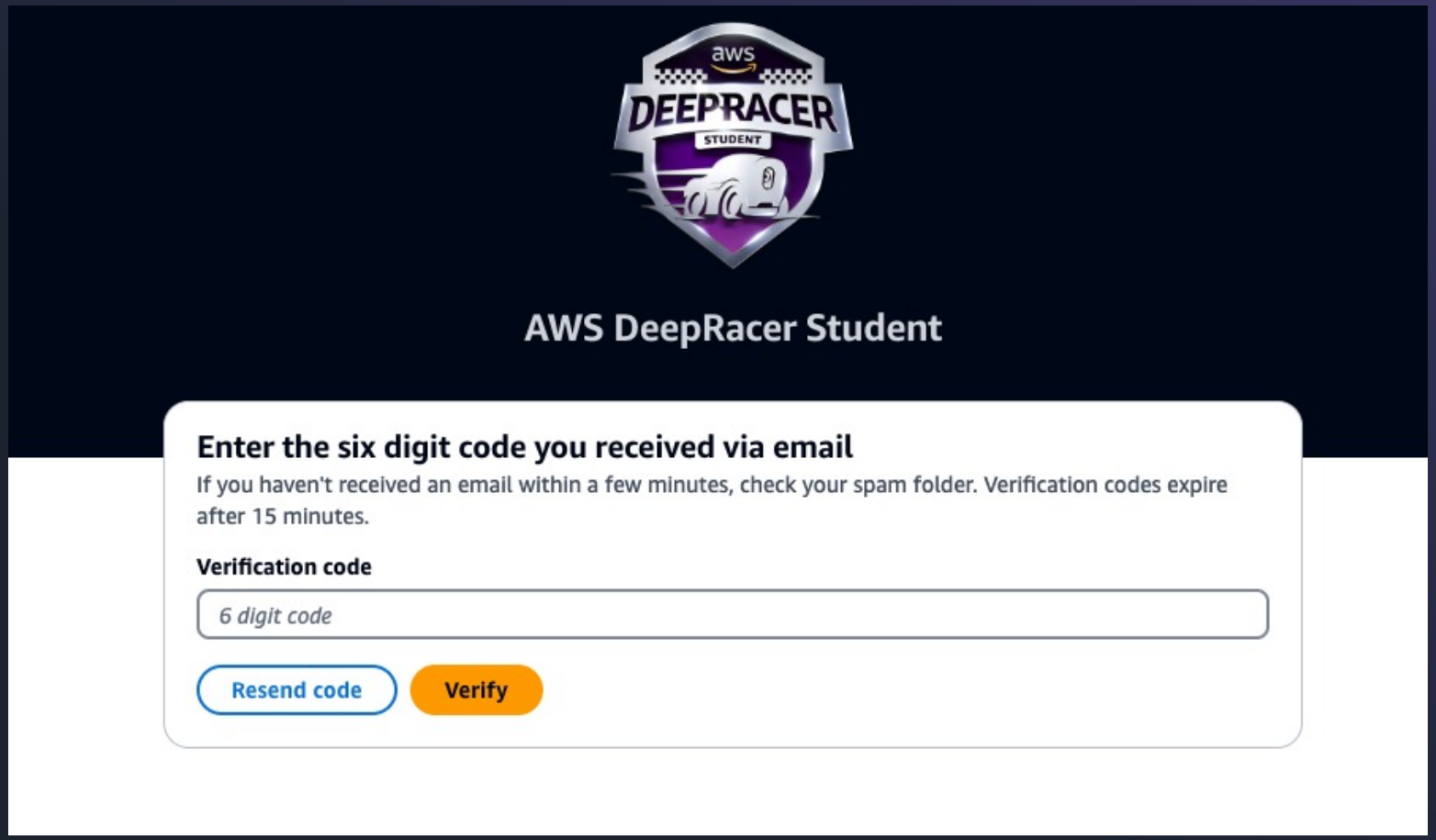


Demo – Create your Student League Account

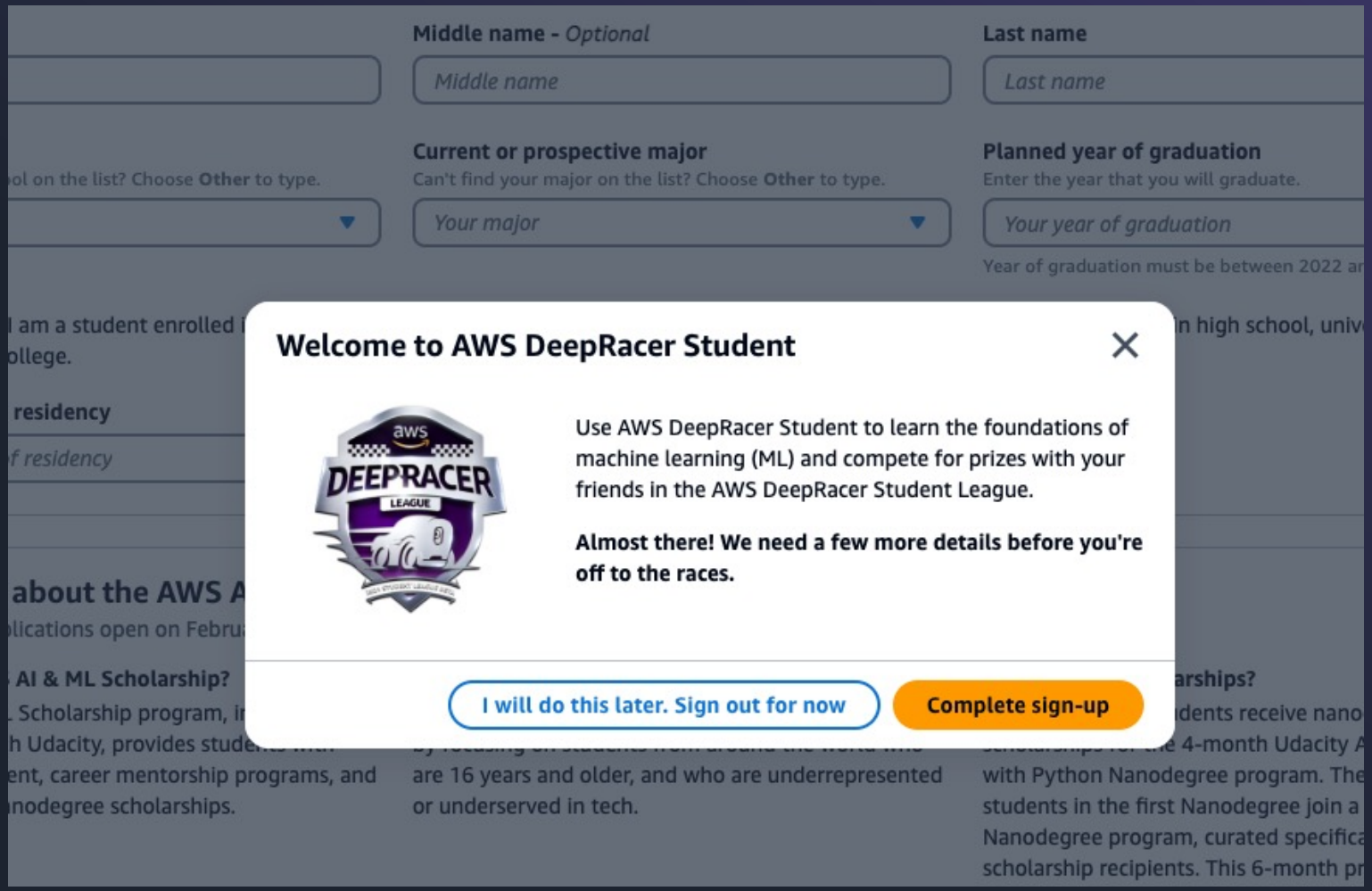
Sing-Up



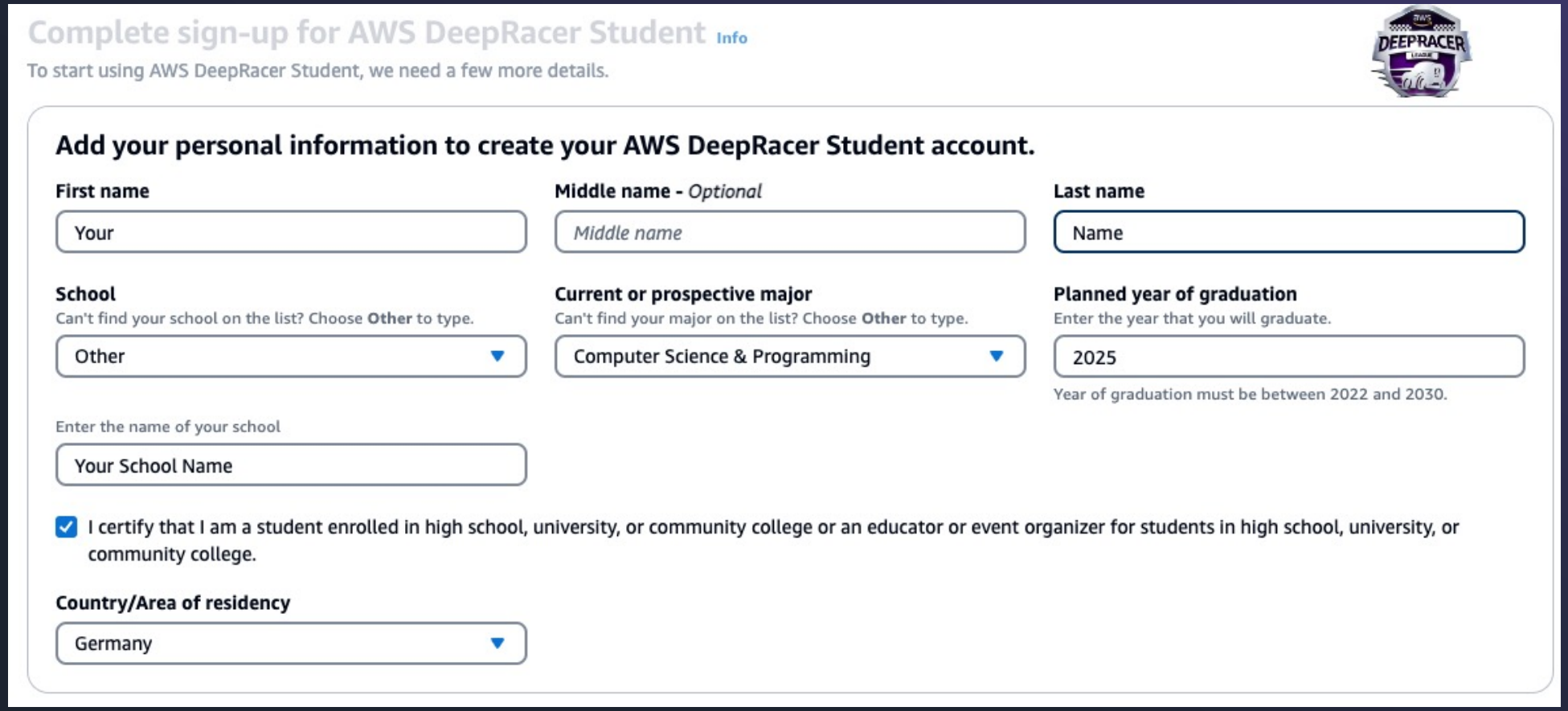
Provide the confirmation Code



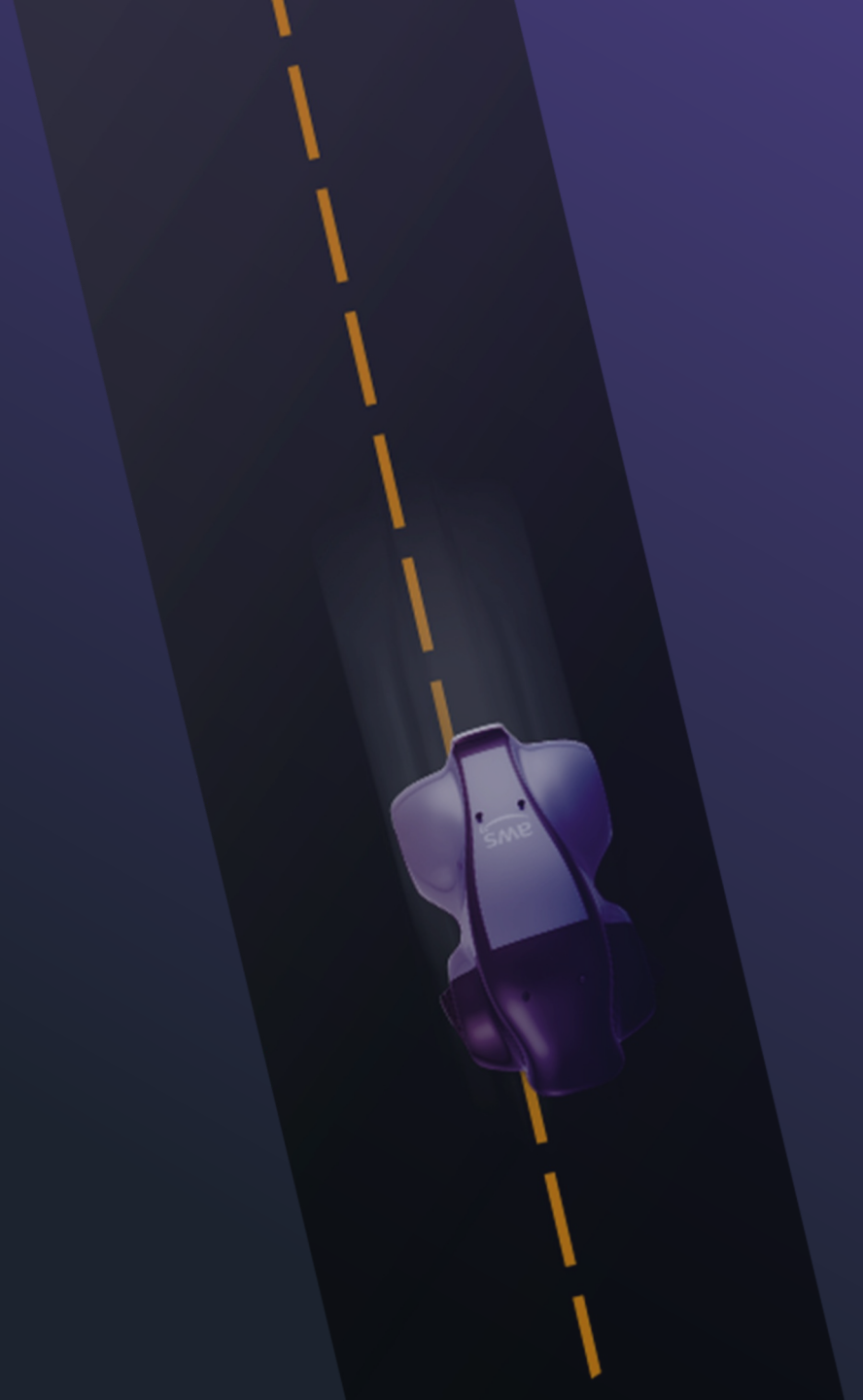
After Sign-In complete Sign-Up



Sign-Up completition



Demo – Training using the AWS DeepRacer console



Next Steps…

AWS DeepRacer League

AWS DeepRacer Student League

Summary

- Introducing AWS DeepRacer
- Introduction to machine learning
- Reinforcement learning key terms
- The reward function
- AWS DeepRacer ML architecture
- Demo

General Points

- Qualifier via Student Accounts
  https://student.deepracer.com/raceToken/s6FunlWBQSOxcptCp7bXkQ
- Track is re:Invent 2018
- Fastest Lap wins
- Direction is counter clockwise

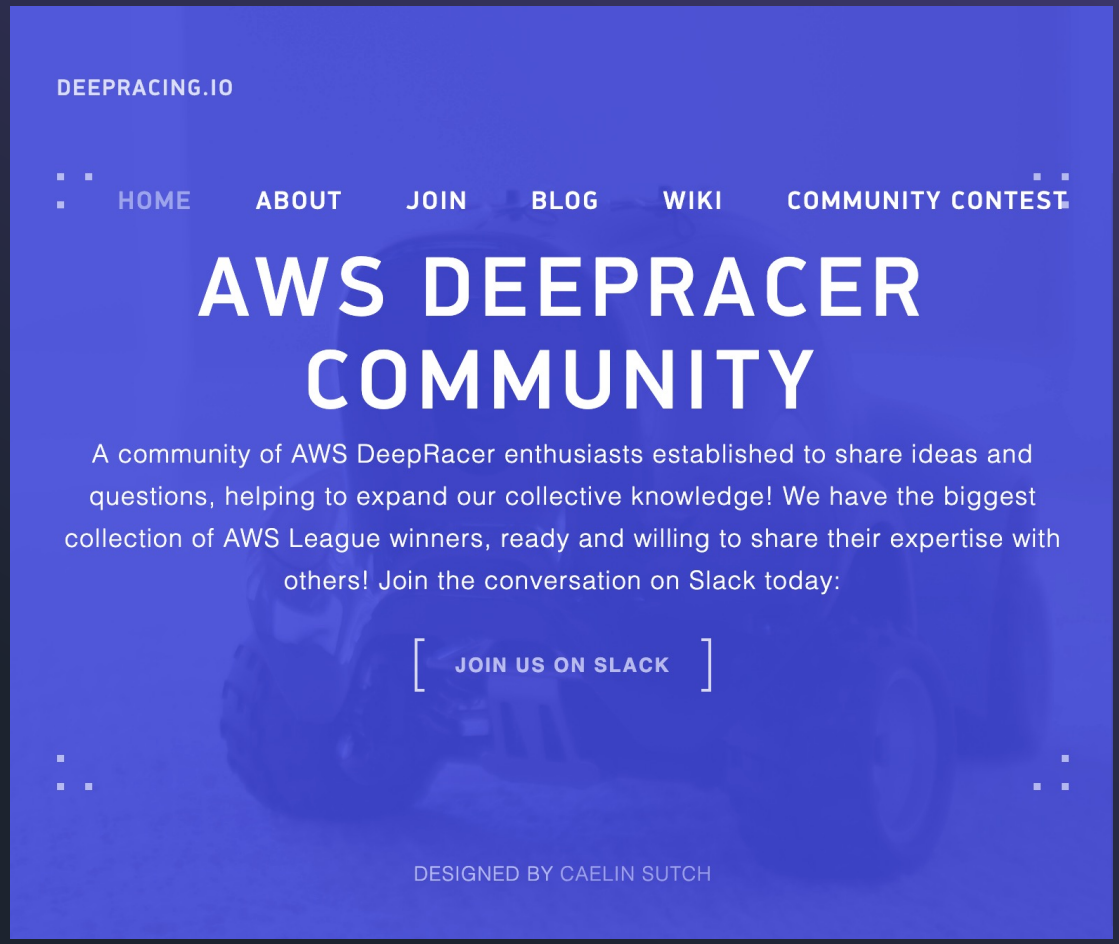
- Models for Live Race can be trained via Student Accounts, regular Accounts or DeepRacer 4 Cloud

Race Day – Racing Rules

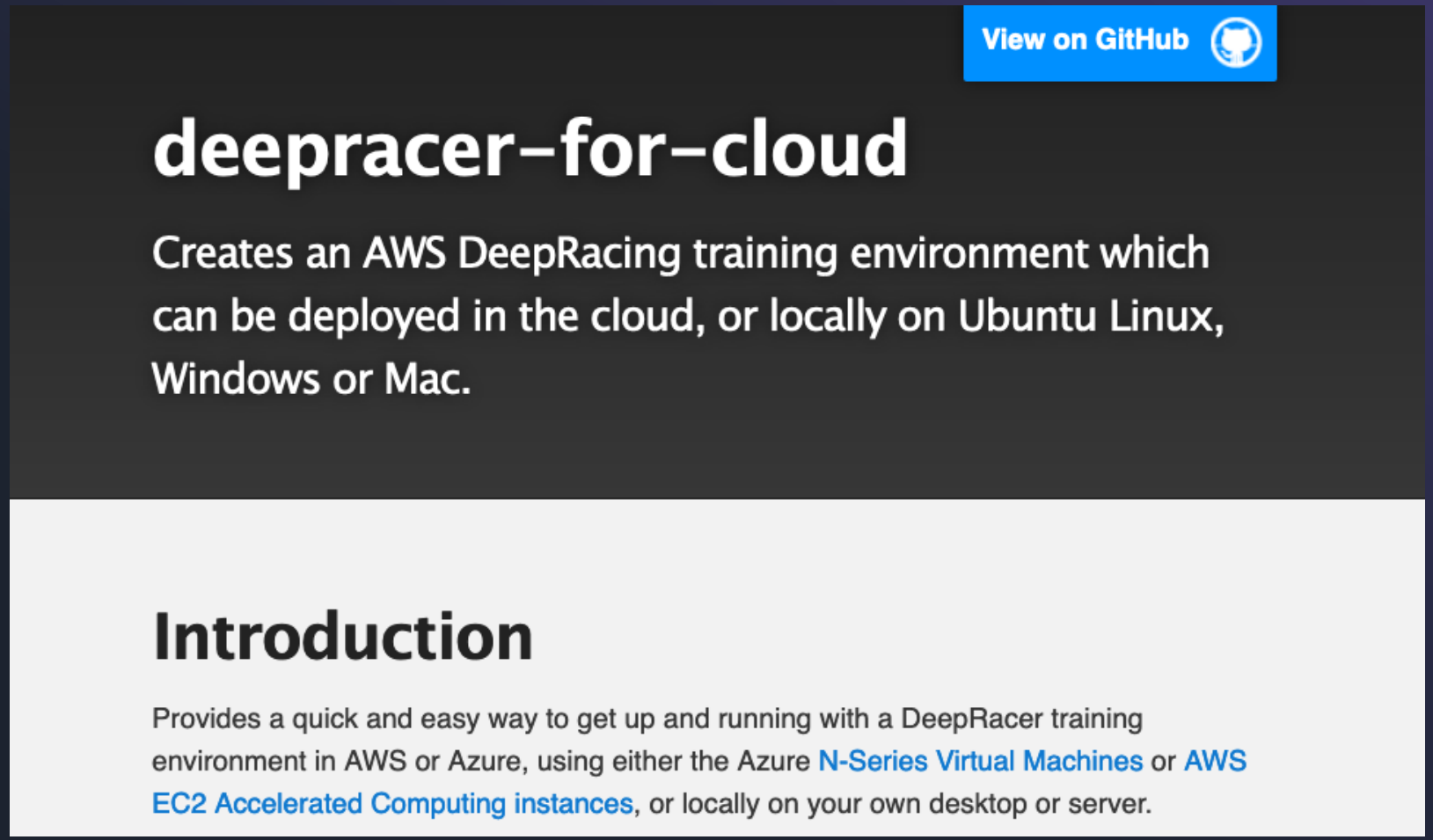
- Track is re:Invent 2018
- Direction is counter clockwise
- 2 minutes per run
- Fastest Lap wins
- Unlimited resets
- Race as much runs as you like and we have time. After a run you have to requeue
- You can race during the whole day

Most Important Ressource

https://deepracing.io/



DeepRacer for Cloud



Resources I

DeepRacer Community:

https://deepracing.io/

Blogs:

https://towardsdatascience.com/an-advanced-guide-to-aws-deepracer-2b462c37eea

https://blog.gofynd.com/how-we-broke-into-the-top-1-of-the-aws-deepracer-virtual-circuit-573ba46c275

https://towardsdatascience.com/penalizing-the-discount-factor-in-reinforcement-learning-d672e3a38ffe

https://towardsdatascience.com/proximal-policy-optimization-tutorial-part-2-2-gae-and-ppo-loss-fe1b3c5549e8

https://awstip.com/exponentially-improving-at-aws-deep-racer-b07ec116087

Resources II

Videos:

https://youtu.be/v7SXfYSWvBU - Log Analysis with Breadcentric

https://www.youtube.com/@BoltronRacingTeam – Bolton Racing Team Channel

DeepRacer for Cloud:

https://aws-deepracer-community.github.io/deepracer-for-cloud/

https://docs.google.com/document/d/1m14DWQPau6ee093i8ebV8iE7wiNwXWy9ALll6dZrIZY/edit#

DeepRacer Log Guru

https://github.com/aws-deepracer-community/deepracer-log-guru

Resources III

Github:

https://github.com/cdthompson/deepracer-k1999-race-lines Racline calculation

https://github.com/aws-deepracer-community/deepracer-race-data/tree/main/raw_data/tracks Tracks

Thingiverse:

https://www.thingiverse.com/thing:5653115 Suspension Tuning part

Let’s hit the road