REWARD FUNCTION

Craft a Powerful Reward Function for AWS DeepRacer Student League

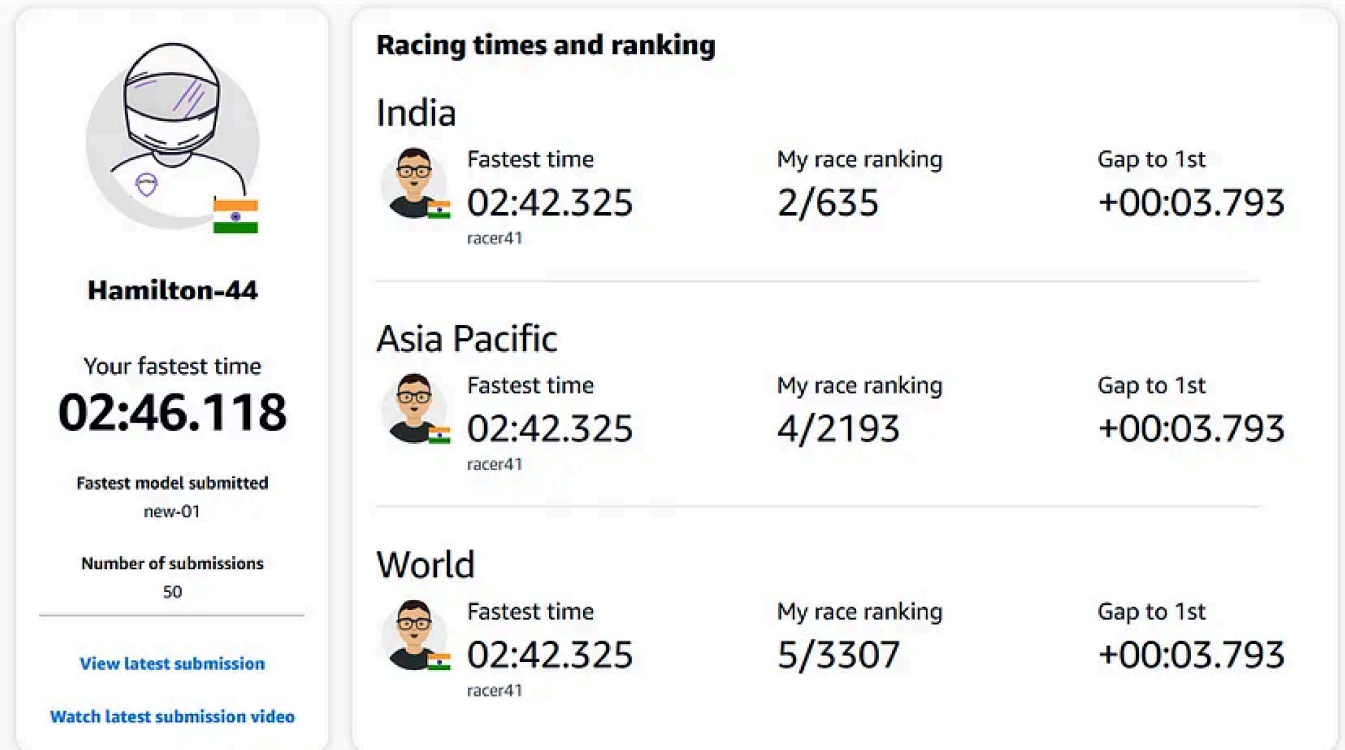
Dominate the Student League of AWS DeepRacer Part-2

Ansh Tanwar · Follow

9 min read · Jun 30, 2023

Listen Share More

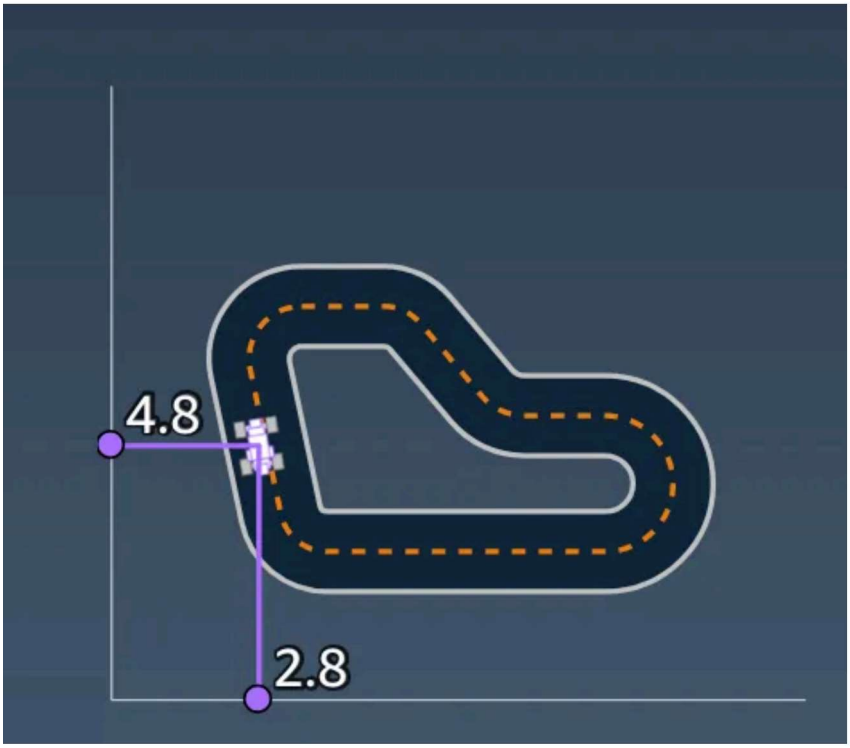
While most articles focus on virtual circuits, I'm here to simplify things for you in the student league and share some tips that I've gathered from the Discord community and other resources available online. These methods have helped me achieve an impressive top 8 rank in the previous season and a top 5 position Globally this season. 🏁

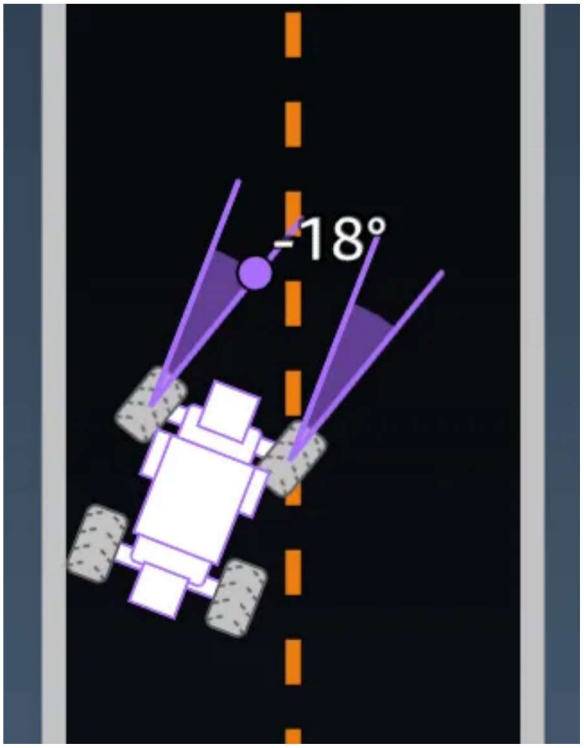


Before crafting let's look at all the parameters that can be used in the reward function

```
    "all_wheels_on_track": Boolean,        # flag to indicate if the agent is c
    "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 clos
    "closest_waypoints": [int, int],       # indices of the two nearest waypoir
    "distance_from_center": float,         # distance in meters from the track
    "is_crashed": Boolean,                 # Boolean flag to indicate whether t
    "is_left_of_center": Boolean,          # Flag to indicate if the agent is c
    "is_offtrack": Boolean,                # Boolean flag to indicate whether t
    "is_reversed": Boolean,                # flag to indicate if the agent is c
    "heading": float,                      # agent's yaw in degrees
    "objects_distance": [float, ],         # list of the objects' distances in
    "objects_heading": [float, ],          # list of the objects' headings in c
    "objects_left_of_center": [Boolean, ], # list of Boolean flags indicating w
    "objects_location": [(float, float),], # list of object locations [(x,y), .
    "objects_speed": [float, ],            # list of the objects' speeds in met
    "progress": float,                     # percentage of track completed
    "speed": float,                        # agent's speed in meters per seconc
    "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

    }
```





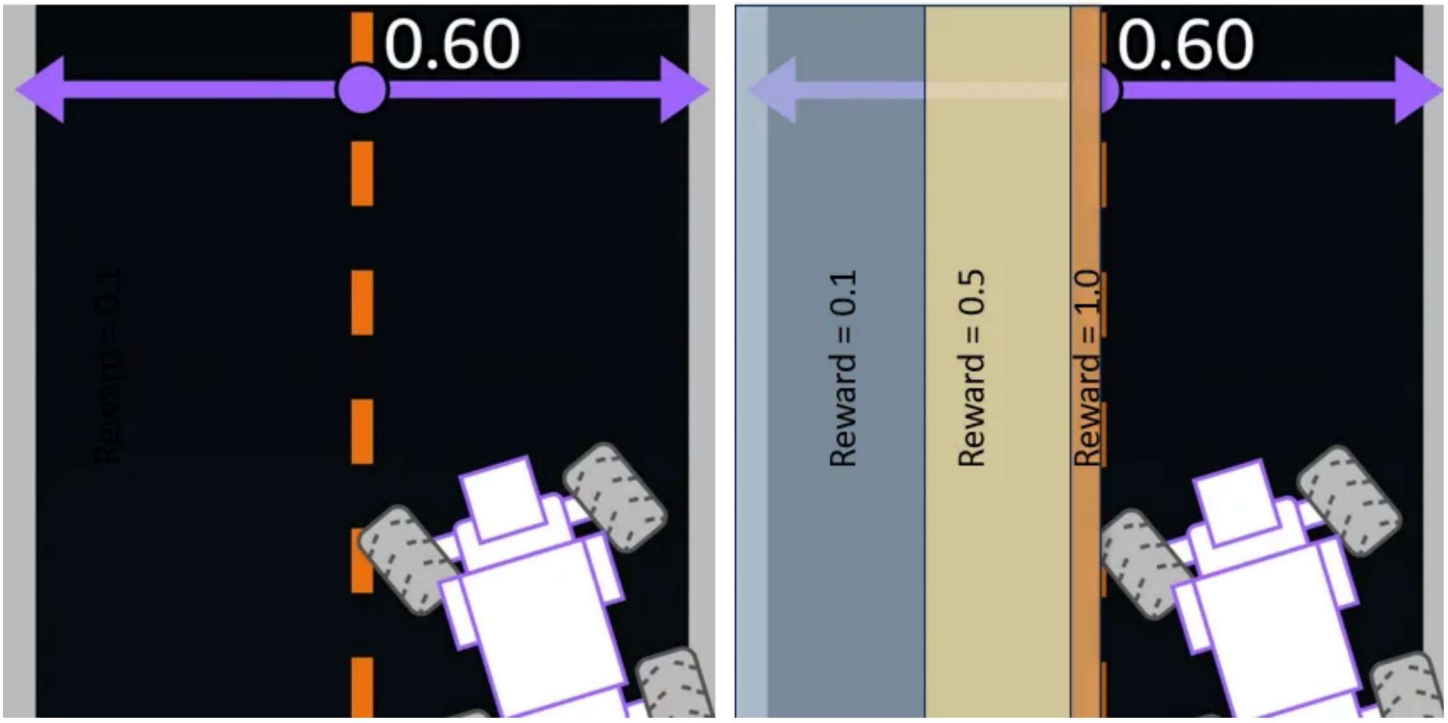
> *To get a more detailed reference of the input parameters along with pictures and examples read this beautiful documentation.*
>
> *link -> Input parameters of the AWS DeepRacer reward function — AWS DeepRacer (amazon.com)*

First, I will walk you through one of the default reward functions provided by AWS and show you how to analyze, improve, and optimize it step by step. You will also discover some tips and tricks on how to increase the speed of your car and avoid common pitfalls. By the end of this article, you will have a solid understanding of how to design your own reward function and achieve top ranks in the AWS DeepRacer League. Let's get started! 😊

```
'''
Example 3: Prevent zig-zag in time trials
This example incentivizes the agent to follow the center line but penalizes
with lower reward if it steers too much, which helps prevent zig-zag behavior.
The agent learns to drive smoothly in the simulator and likely keeps
the same behavior when deployed to the physical vehicle.
'''

def reward_function(params):
    '''
    Example of penalize steering, which helps mitigate zig-zag behaviors
    '''

    # Read input parameters
    distance_from_center = params['distance_from_center']
    track_width = params['track_width']
    abs_steering = abs(params['steering_angle']) # Only need the absolute steer

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

    # Give higher reward if the car is closer to center line and vice versa
    if distance_from_center <= marker_1:  # REGION 1
        distance_reward= 1.0
    elif distance_from_center <= marker_2: # REGION 2
        distance_reward= 0.5
    elif distance_from_center <= marker_3: # REGION 3
        distance_reward= 0.1
    else:
        distance_reward= 1e-3  # likely crashed/ close to off track
                               #never set negative or zero rewards

    # Steering penality threshold, change the number based on your action space
    ABS_STEERING_THRESHOLD = 15

    # Penalize reward if the car is steering too much
    if abs_steering > ABS_STEERING_THRESHOLD:
        distance_reward*= 0.8

    return float(reward)
```

In this we are calculating the distance from the center of the track and giving higher reward if the car is closer to center line.

- `marker_1` is set to 10% of the track width .
- `marker_2` is set to 25% of the track width.
- `marker_3` is set to 50% of the track width.



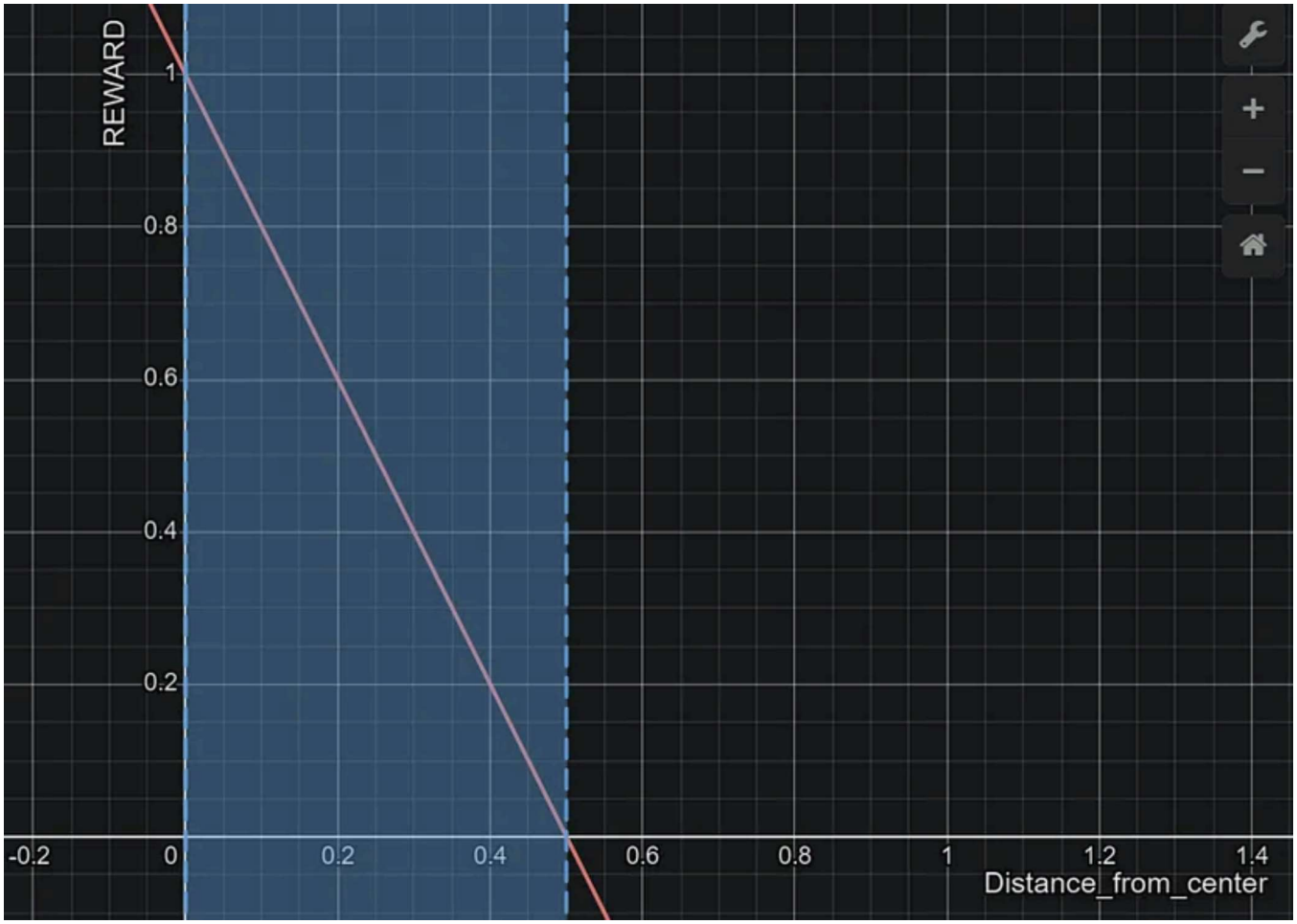
explains the different regions on the track and the rewards associated with them

we can experiment by varying ***marker length*** and ***reward values*** but lets think differently 🧠.

Lets give a **continuous reward** instead of discreate.

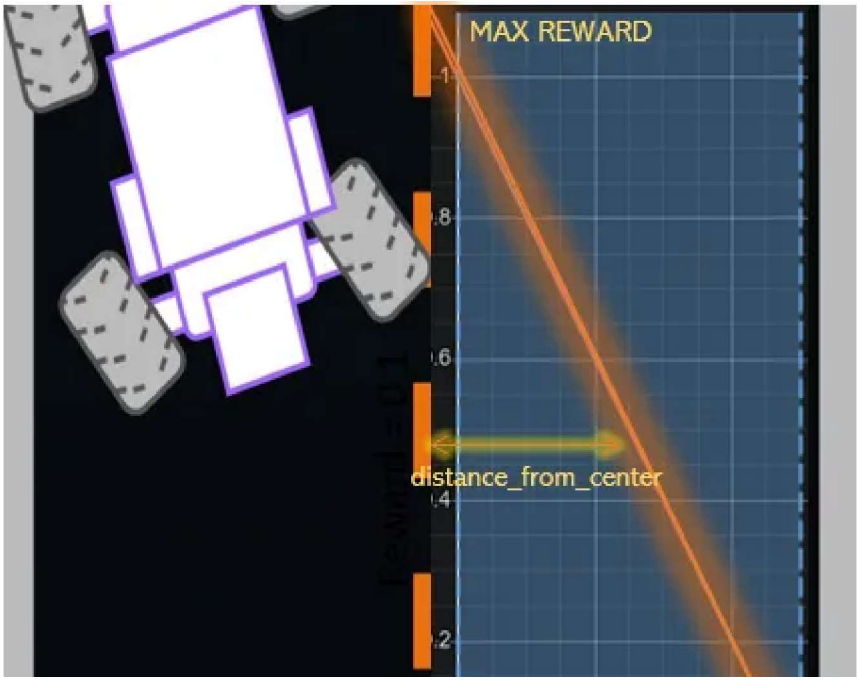
```
#continous reward
distance_reward = 1- (distance_from_center/track_width*0.5)
```

> *y= reward*
> *x = distance from center , where 0.0 < x < 0.5 m*
>
> *let's take track_width = 1.0 m for simplicity.*



made using desmos

The reward function is a linear function of the distance from the center of the track. The closer the car is to the center, the higher the reward. The car gets zero reward if it goes off track (distance from center > 0.5 m).

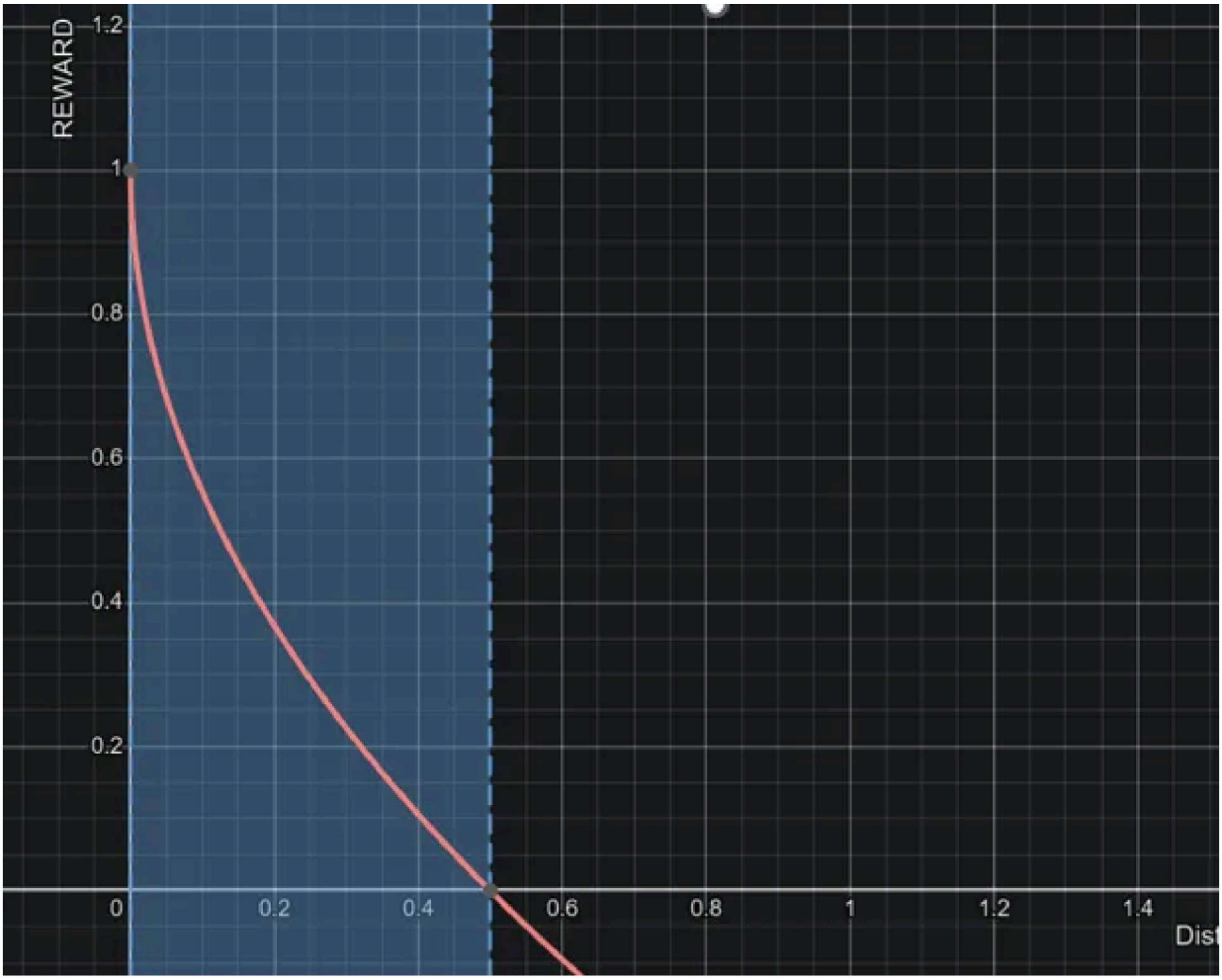


The image below show's how reward changes with different positions of car on the track.

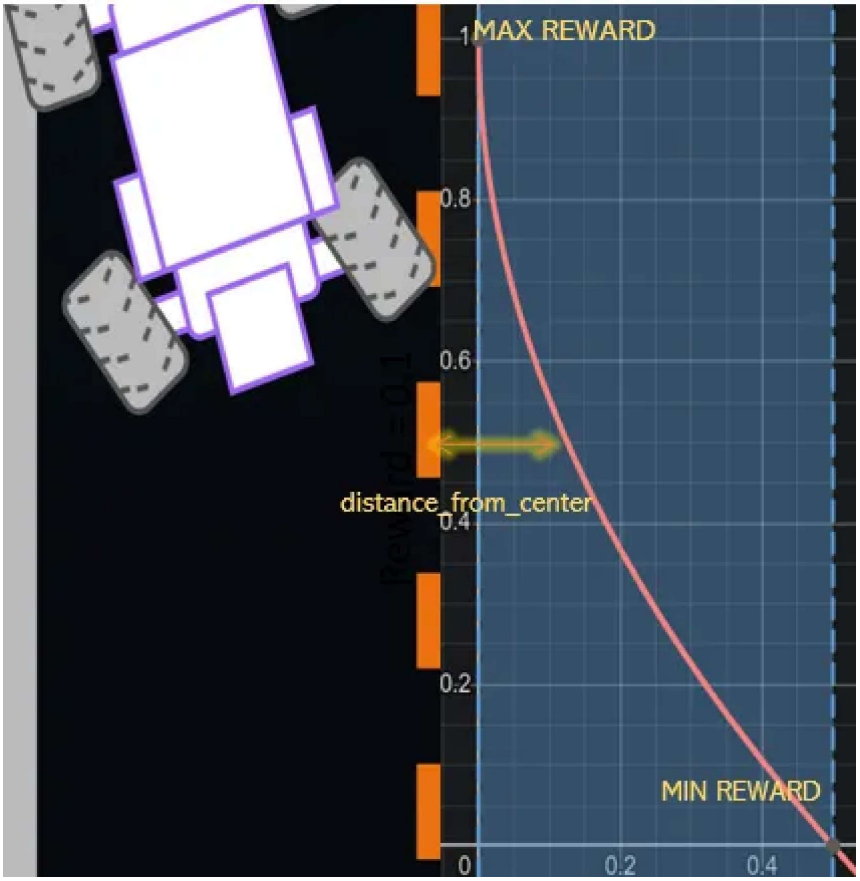
A continuous reward offers fine-grained feedback, smooth learning, proportional rewards, gradient information, and customizability. These benefits contribute to more precise learning, improved convergence, and better performance of the reinforcement learning agent.

> *Now lets assume you want the car to stay more towards center but also give a gradual reward otherwise, so we can use an exponential function like this*

```
distance_reward = 1- (distance_from_center/track_width*0.5)**0.4
#we haveraised to the power 0.4
#to make the curve more step lower the value
```


REWARD
1.2
1
0.8
0.6
0.4
0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4

$y = 1-(x/0.5)^{4}/^{10}$


MAX REWARD
1
0.8
0.6
0.4
0.2
0
0.2
0.4
distance_from_center
MIN REWARD

The value of our reward will vary between 0 — 1.0. Also we can scale the value of distance reward by seeing other parts of our complete reward function.

```
#Example
final_reward = distance_reward*2 + speed_reward*10 + ......
```

MAINTAINING SPEED OF 1 M/S

One of the key challenges in AWS DeepRacer is to maintain a speed of 1.0 m/s throughout the race. This can give you a significant advantage over your competitors and help you achieve better results.So lets dive deep to optimize our speed performance.

Even if I share my reward function with you, it will not work for you or your track. Each track requires different methods to maintain the speed. That's why I want to share the method and ingredients that you will need to build your own.

Just using the speed parameter and rewarding the car for it will not do the work. Maintaining the speed requires careful and skillful building of the complete reward function, such that each of its subcomponents contributes towards maintaining the best speed overall.

```
def reward_function(params):
    ################################################################################
    '''
    Example of using all_wheels_on_track and speed
    '''

    # Read input variables
    all_wheels_on_track = params['all_wheels_on_track']
    speed = params['speed']

    # Set the speed threshold based your action space
    SPEED_THRESHOLD = 1.0

    if not all_wheels_on_track:
        # Penalize if the car goes off track
        reward = 1e-3
    elif speed < SPEED_THRESHOLD:
        # Penalize if the car goes too slow
        reward = 0.5
```

```
    else:
        # High reward if the car stays on track and goes fast
        reward = 1.0

    return float(reward)
```

Look at this function in the documentaion Input parameters of the AWS DeepRacer reward function — AWS DeepRacer (amazon.com)

Lets reward and penalize our agent more.

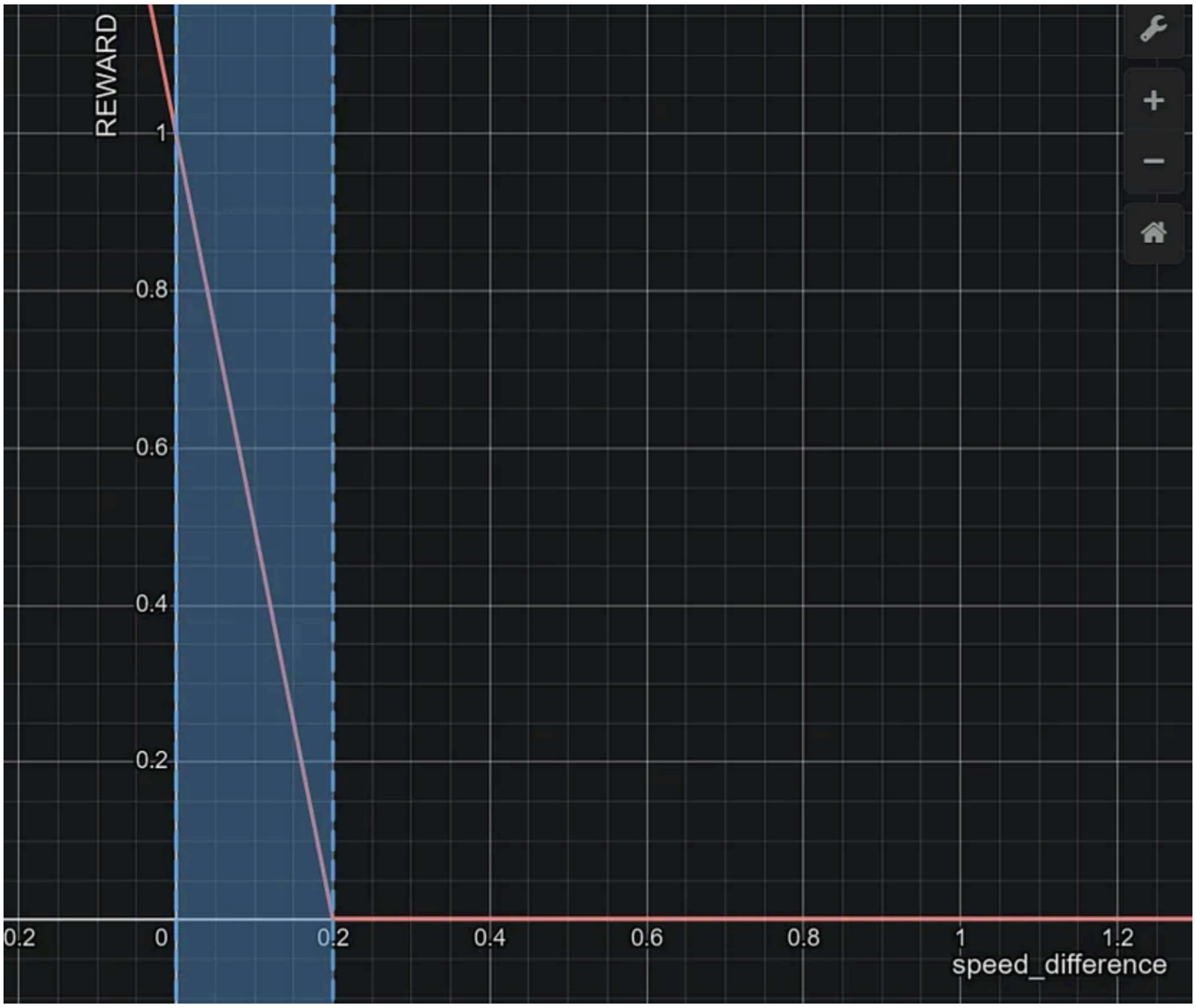
```
if not all_wheels_on_track:
        # Penalize if the car goes off track
        reward = 1e-3 #never set negative or zero rewards
    elif speed < SPEED_THRESHOLD:
        # Penalize more
        reward = 0.1
    else:
        # reward more
        reward = 10.0

'''

The SPEED_THRESHOLD is fixed to 1.0.
This is the maximum speed of the car.
One thing we can try is to change the reward values.

Range to give the rewards [1e-3,1e+3]
keep the scale similar to other subcomponents in your function
The final reward in end can be normalized

'''
```

This can give a significant boost to our speed. You might be wondering why we aren't giving continuous reward for speed too. If so, you are on the right track.

Lets explore continuous speed reward functions

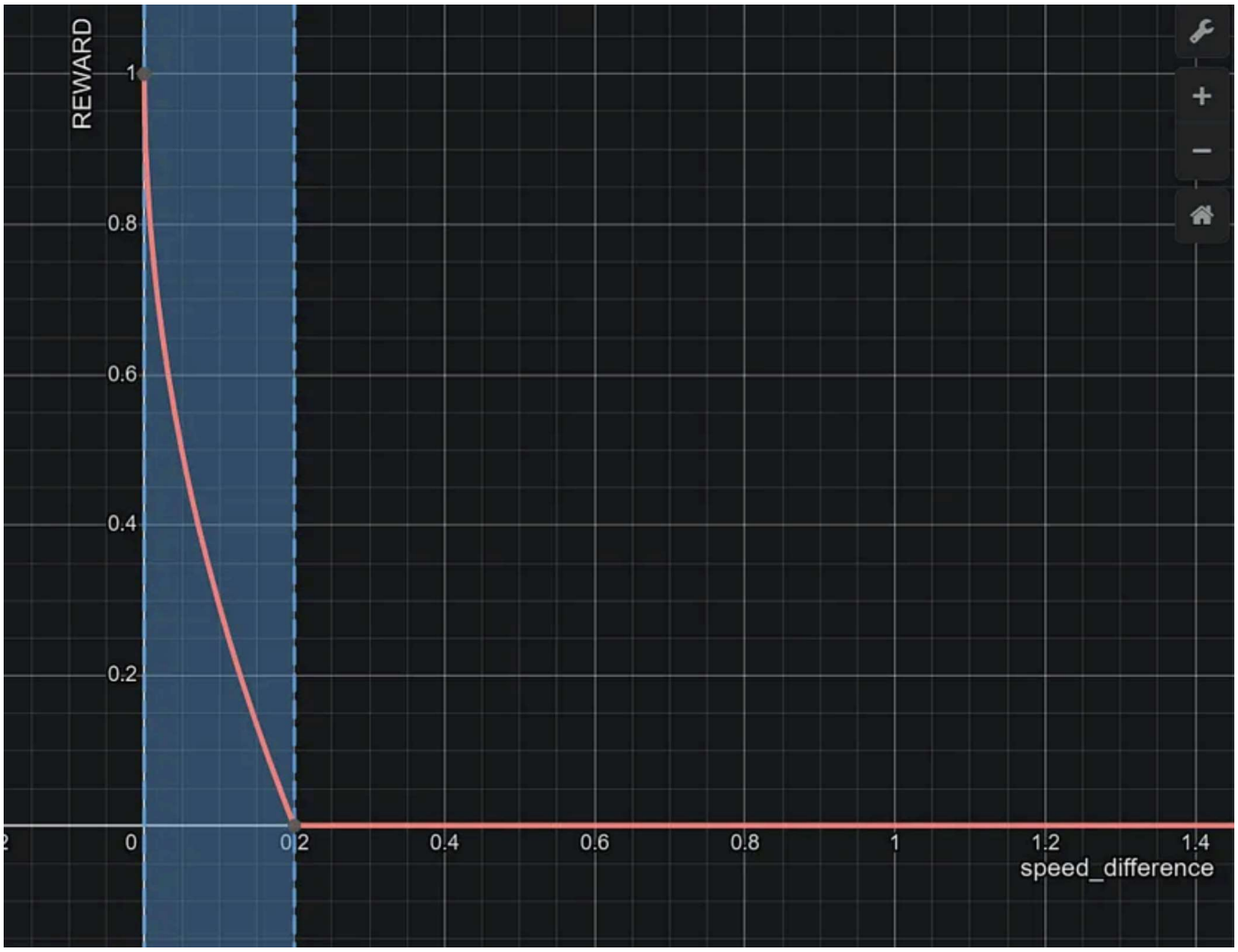
```
speed_diff = abs(1.0-speed)
max_speed_diff = 0.2#set it carefully in range [0.01,0.3]
```

```
if speed_diff < maximum_speed_diff:
  speed_reward = 1-(speed_diff/max_speed_diff)**1
else:
  speed_reward = 0.001 #never set negative or zero rewards
```



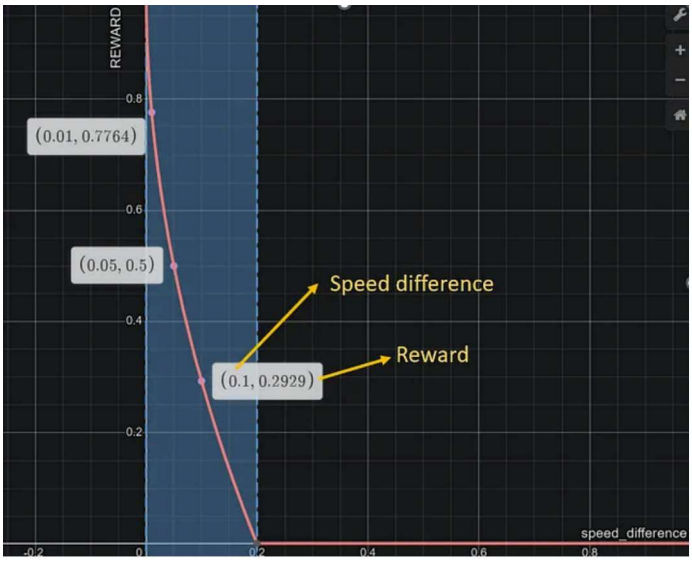

```
'''

lets assume max_speed_diff = 0.2
y=speed_reward
x=speed_difference 0<x<1.0

'''
speed_reward = 1-(speed_diff/max_speed_diff)**0.5
```

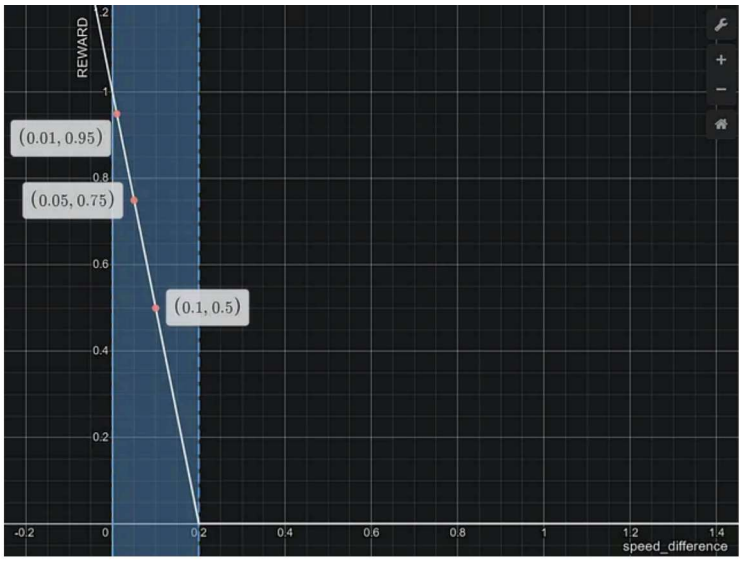
```
    # changing the shape of the curve from linear to exponential helps a lot
```

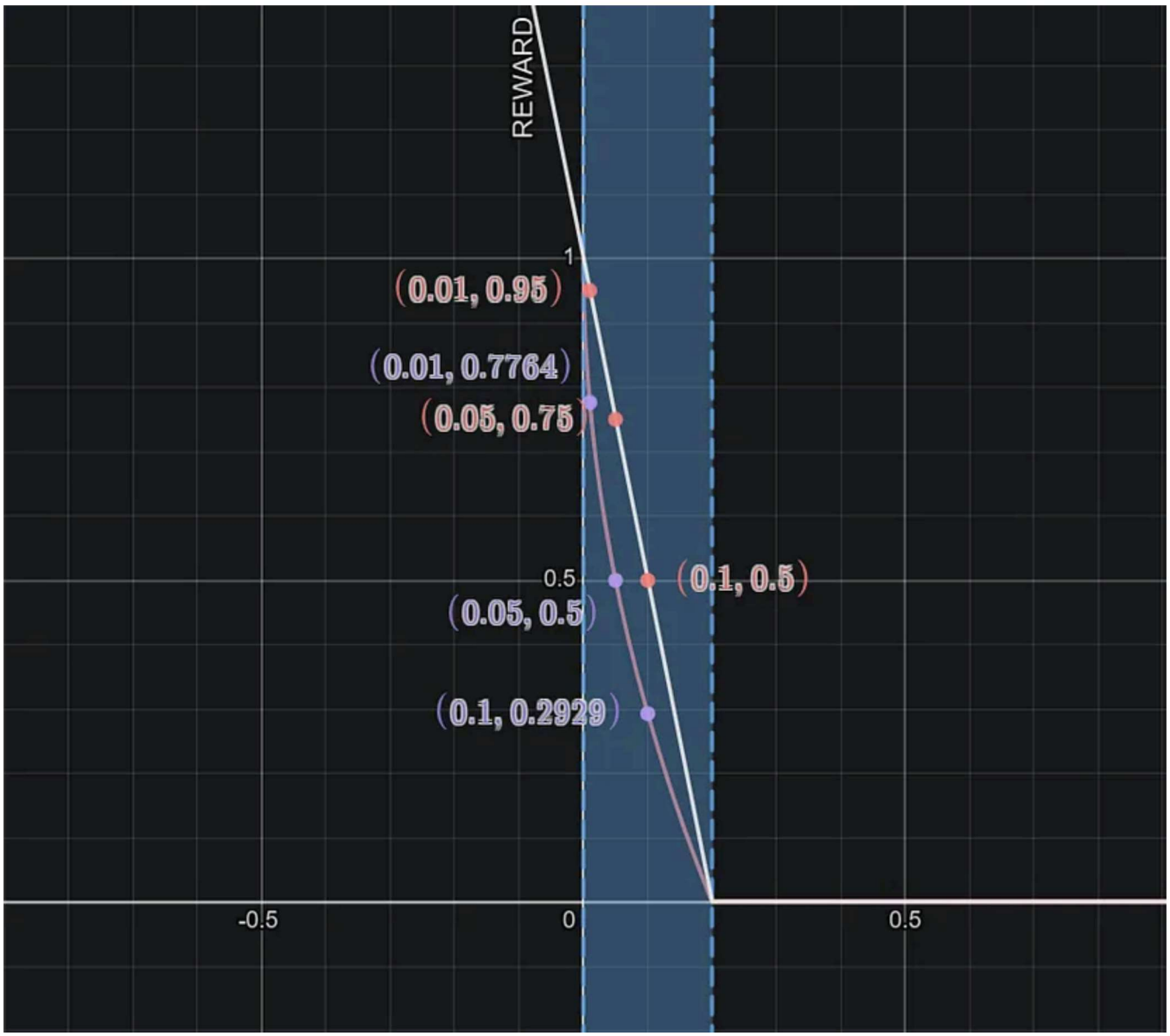


Notice how much reward the agent gets for both the shapes/slopes for :-

> *speed difference (1.0 — speed of car) = 0.1, 0.05 and 0.01 i.e.*
>
> *speed of car = 0.9 and 0.95 m/s and 0.99m/s*







For less speed you get lesser rewards and as you progress higher in speed i.e., closer to 1m/s, the reward increases exponentially high, which motivates the agent to maintain high speed.

Another way to modify the above reward function is to use max() because we are not using any if condition and we don't want our reward to go below 0.001.

```
max_speed_diff = 0.2
speed_diff = abs(1.0-speed)
    speed_reward = max(1e-3, 1-((speed_diff/max_speed_diff)**0.5 ))
```

Steps and Progress components are also very important to maintain that 1 m/s speed

let's look at the example given in documentation.

```
def reward_function(params):
    ################################################################################
    '''
    Example of using steps and progress
    '''

    # Read input variable
    steps = params['steps']
    progress = params['progress']

    # Total num of steps we want the car to finish the lap, it will vary depenc
    TOTAL_NUM_STEPS = 300

    # Initialize the reward with typical value
    reward = 1.0

    # Give additional reward if the car pass every 100 steps faster than expect
    if (steps % 100) == 0 and progress > (steps / TOTAL_NUM_STEPS) * 100 :
        reward += 10.0

    return float(reward)
```

This function can also be modified and improved in various ways, but the one presented above works very well if accompanied by other subcomponents like speed, distance, etc.

But how do we combine them to make a single reward function?

Here, your knowledge and concepts of reinforcement learning and mathematics play a crucial role. Let me explain to you one of such simple techniques that might work for you.

The Recipe

```
'''
->  Dont set a global variable for reward value
instead set different rewards for each part paramter - speed_reward, steps_rewa

->  Keep the scale for each reward part same, normalizing them helps sometimes

->  Simply adding them together in final_reward variable usually works

->  you can experiment with a lot of methods here, I will strongly
recommend to watch youtube video- https://www.youtube.com/watch?v=vnz579lSGto&a
'''
```

SUMMARY

In this article, you have learned how to craft a powerful reward function for AWS DeepRacer in Student League. You have learned what AWS DeepRacer is, what a reward function is, and how to design and improve your own reward function using different methods and subcomponents. Remember this is just a starting point and you can go a way deeper inside the deepracer . Keep reading more articles on medium and following more resources.

I hope you found this article helpful and informative.

Must Visit :

> *https://medium.com/@anshml/top-tips-for-students-to-dominate-the-aws-deepracer-student-league-eaecde6e3d33*
>
> *Best reward function shapes for success! — YouTube*
>
> *Input parameters of the AWS DeepRacer reward function — AWS DeepRacer (amazon.com)*
>
> *Parameters In Depth: DeepRacer Student League Guide | by Aleksander Berezowski | Medium*
>
> *Results and Lessons: DeepRacer Student League March 2023 | by Aleksander Berezowski | Medium*

If you liked this article, please give it some claps 👏 and share it with your friends who are interested in AWS Deep Racer. If you have any doubt do ask in comment section.

Also, don't forget to follow me for more articles on AWS DeepRacer and other topics related to machine learning and data science.

Thank you for reading and happy racing! 😊

AWS | Aws Deepracer | Deepracer | Machine Learning | Reinforcement Learning

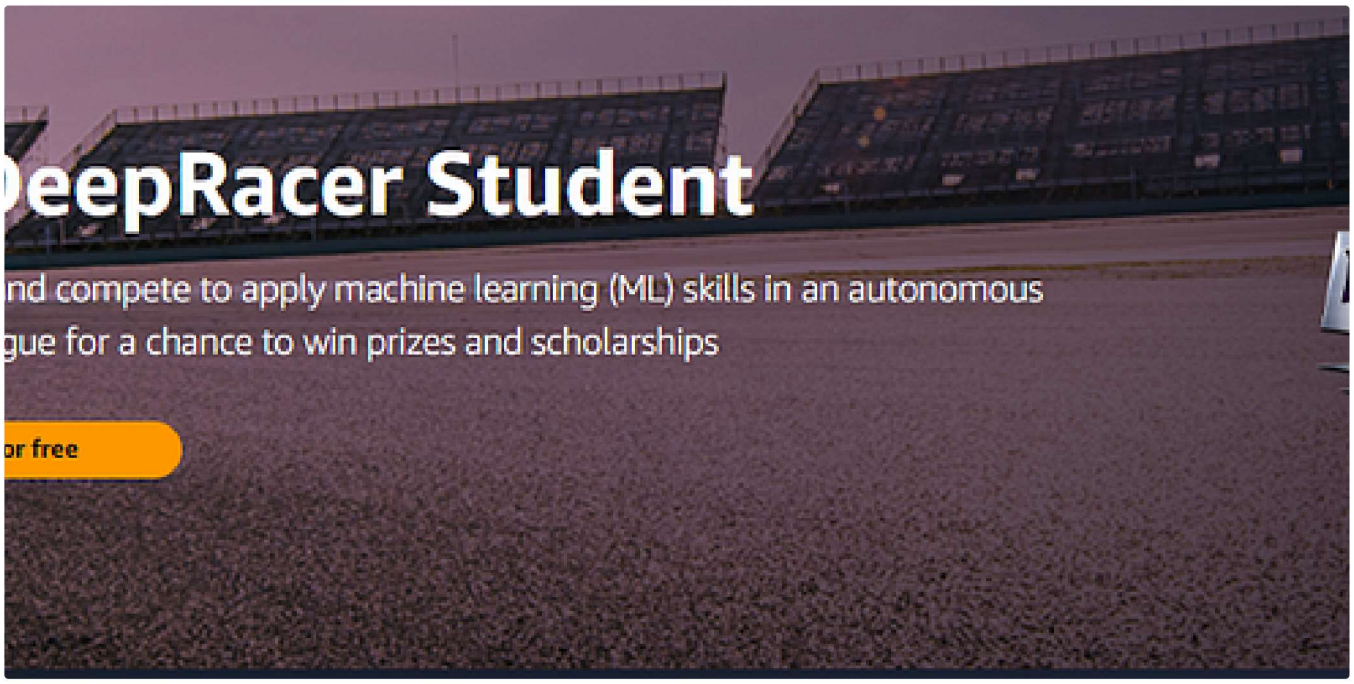
Follow

Written by Ansh Tanwar

65 Followers

Talks about AI/ML and Technology.

More from Ansh Tanwar



Ansh Tanwar

Dominate the Student League of AWS DeepRacer

PART-I

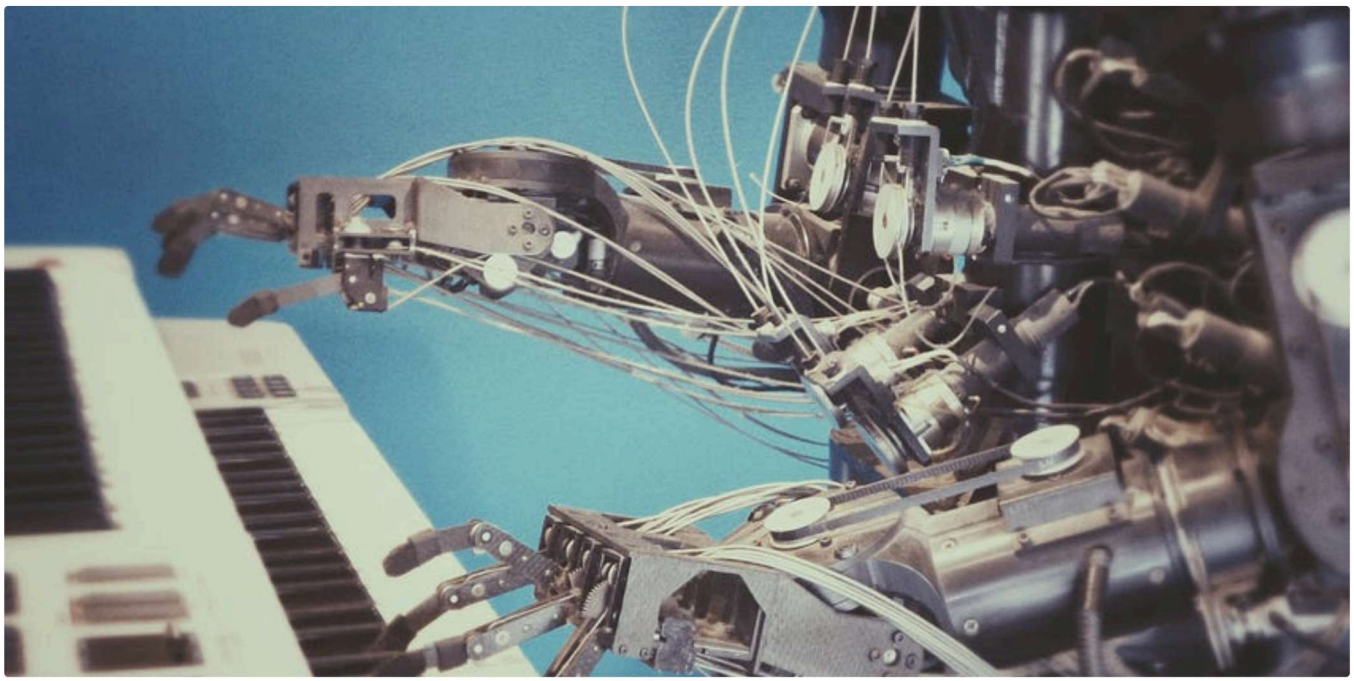
Mar 24, 2023 179 3

Ansh Tanwar

AWS AI & ML Scholarship Program worth $4,000 USD and Mentorship

Are you interested in learning more about artificial intelligence (AI) and machine learning (ML)? Do you want to pursue a career in this...

Aug 6, 2023 86 1



Ansh Tanwar

🤖 Top 10 AutoML and AutoEDA Libraries with Starter Notebooks.

AutoML and AutoEDA are two of the most popular and useful Python techniques for data science and machine learning. They allow you to...

Aug 6, 2023 87 3

Ansh Tanwar

The Thrilling Fusion of Reinforcement Learning and AWS DeepRacer

Have you ever wondered how self-driving cars learn to navigate complex environments? How robots learn to perform tasks that humans take for...

Jan 10 97 1

See all from Ansh Tanwar

Recommended from Medium

Building Personal Projects / Resume	Try building larger scale, prevalent projects	Build upon past projects, finish them up, or build smaller but more impactful projects	Restructure and Finalize Resume	
Networking	Attend Conferences and Career Fairs, Contact University Recruiters	Follow up on your conversations, ask for 1-1s with Recruiters	Continue building relationships	Continue building relationships
Building Online Presence	Document your projects through writing or videos	Keep updating your LinkedIn		
Leetcode Intensity	Review and Understand DSA	Aim for 1/day	Aim for 2/day	Aim for 2/day Redo problems Mock Interview

Rimika Dhara in Women in Technology

Ultimate Timeline for Landing a Summer 2025 SWE Internship

How you can prepare now to land your dream summer internship in summer 2025. Short and simple.

Jun 17 54

Mikel.D

How I Thrived as a Machine Learning Engineer Intern at Apple AI-ML HQ.

Insider Tips, Pros, and Cons for Interview and Internship Success!

Apr 29

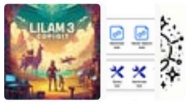
Lists

Predictive Modeling w/ Python

20 stories · 1343 saves

Practical Guides to Machine Learning

10 stories · 1623 saves



Natural Language Processing

1563 stories · 1104 saves

The New Chatbots: ChatGPT, Bard, and Beyond

12 stories · 421 saves

Nollie Chen

AWS Internship Week 2 Reflections

I participated in our team's monthly retrospective meeting. Besides our daily stand-ups, we also review monthly highlights and lowlights.....

Jun 17 1

Ushmita Dutta

Amazon ML Summer School Experience

Round 1: Online Assessment

Jun 11 9 1

Praveen Sambu in AWS in Plain English

What is Amazon Q?

A new generative AI powered Assistant from AWS. Amazon Q will revolutionize the way the business , builders, developers will interact with...

Mar 26 50

Kevin Shan in DataDrivenInvestor

The Kurv ETFs That Yield Between 11.36% and 26.05% Selling Synthetic Covered Calls

Can this company compete against YieldMax?

5d ago 61 1

See more recommendations