AWS Academy Learner Lab Activities

Activity - Reinforcement Learning with AWS DeepRacer

Description

AWS Academy Learner Lab Activities are short duration hands-on activities focused on particular topics that can be implemented in the AWS Academy Learner Labs. This activity provides easy-to-follow guidance to build, train and evaluate a reinforcement learning models using AWS DeepRacer.

Learning objectives

Upon completion of this activity, learners should be able to:

- Use AWS DeepRacer to create and train a reinforcement learning model.
- Use AWS DeepRacer to evaluate a trained model.
- Use AWS DeepRacer to enhance the model reward function.

How to use this Activity

This document provides the step-by-step guidance to create and evaluate a reinforcement learning model using AWS DeepRacer in the Academy Learner Lab. AWS Academy Educators can use this activity to explore AWS DeepRacer with their learners during class or share the activity sheet with their learners for a more prescriptive approach.

Intended audience

This project is intended for students who attend AWS Academy member institutions, seek to gain hands-on experience in use cases and functions of machine learning (ML) technology.

Educator and Student prerequisites

This project does not have any prerequisites for educators and learners. However, educators and learners are recommended to complete the AWS Academy Machine Learning Foundations course.

Delivery methods

Learning materials are provided to support in-person or online synchronous delivery.

Duration

The duration will vary depending on how educators integrate this project into course activities and can be from 1 to 2 hours.

Activity Sheet: Activity - Reinforcement Learning with AWS DeepRacer

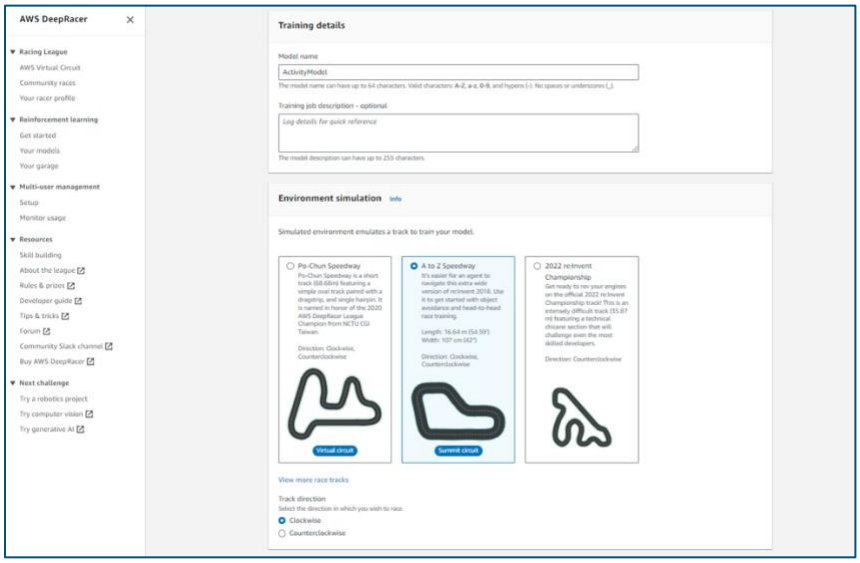
Introduction

In this activity educators and learners will use AWS DeepRacer to apply machine learning concepts. This activity guides educators and learners on how to build, train and evaluate a reinforcement learning models using AWS DeepRacer.

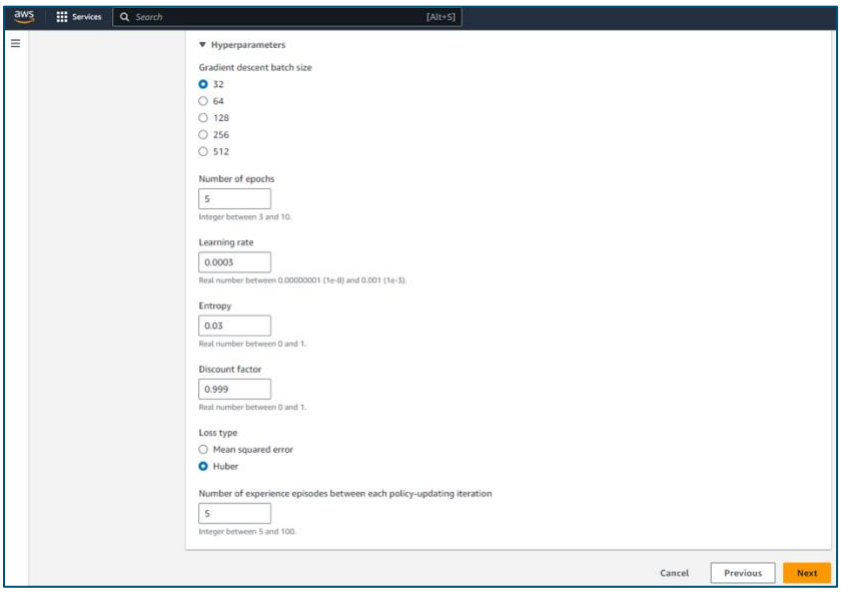
Task 1: Create a model to get started

Follow the next steps to create a model to get started:

1. Access to your AWS Academy Learner Lab class and start the environment.
2. Access to the AWS console.
3. From the **Services** menu, choose **Machine Learning** > **AWS DeepRacer**.
4. In the left navigation pane, choose **Your models**.
5. Choose **Create model**
6. Under **Training details** section enter a name for your model. For example, ActivityModel.
7. Under **Environment simulation** section choose **A to Z Speedway**
8. Select **Clockwise** as the track direction.
9. Choose **Next**



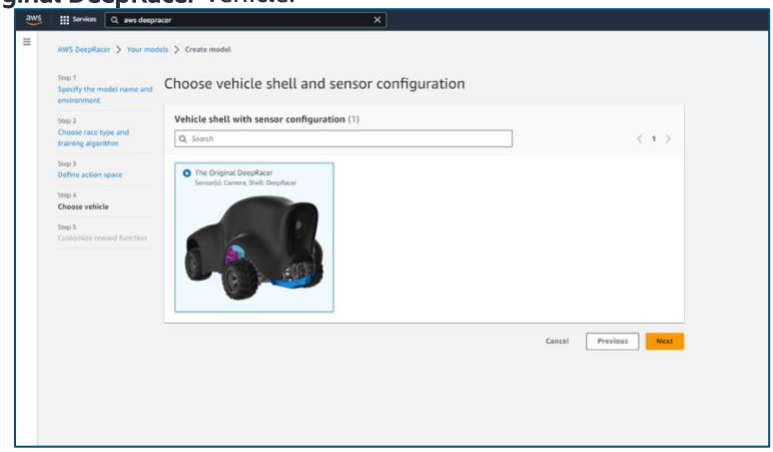
10. Choose **Time trial** as **Race type**. You can learn more about all available race types here
11. Choose **PPO** as **Training algorithm**. You can learn more about training algorithms here
12. Expand the **Hyperparameters** section.
    Note: Adjusting Hyperparameters is an advanced topic. For the purposes of this educational activity and in order to obtain as many data points as possible in short training times, configure the following:
    - Gradient descent batch size: 32
    - Number of epochs: 5
    - Learning rate: 0.0003
    - Entropy: 0.03
    - Discount factor: 0.999
    - Loss type: Huber
    - Number of experience episodes between each policy-updating iteration: 5



13. Choose **Next**
14. Under **Select action space** choose **Discrete action space**.

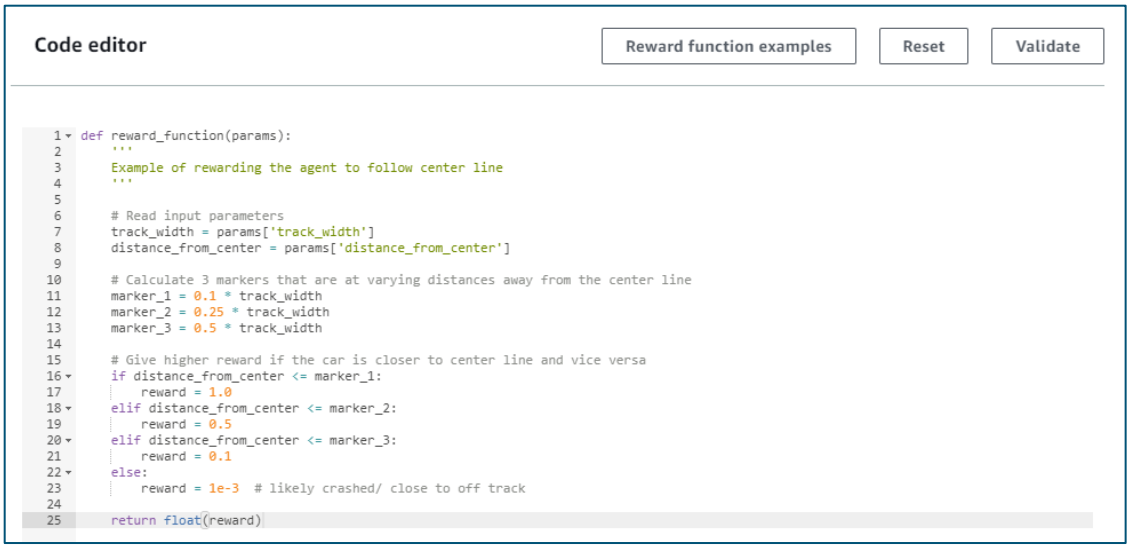
    Note: In reinforcement learning, the set of all valid actions, or choices, available to an agent (car) as it interacts with an environment is called an action space. In the AWS DeepRacer console, you can train agents in either a discrete or continuous action space.
    During training, the car will receive inputs from the environment thru the simulated sensors, and will try all possible action items to learn which one will give it the most reward, both in the short term and the long term. The longer the training, the more combinations (state → action) will be tested. Thus, the action space shapes the number of nodes in the neural network, and as a result, we can't change the number of actions if we want to keep improving the same model.

15. Use the default **Steering angle** and **Speed** values.
    - Steering angle granularity: 5
    - Maximum steering angle: 30
    - Speed granularity: 2
    - Maximum speed: 1
16. Choose **Next**
17. Choose **The Original DeepRacer** vehicle.



18. Choose **Next**.
19. Don't modify the default **Reward function** example.

    Note: The reward function is at the core of reinforcement learning. Learn to use it to incentivize your car (agent) to take specific actions as it explores the track (environment). Like encouraging and discouraging certain behaviors in a pet, you can use this tool to encourage your car to finish a lap as fast as possible and discourage it from driving off of the track or colliding with objects. This example determines how far away the agent is from the center line, and gives higher reward if it is closer to the center of the track, encouraging the agent to closely follow the center line. It uses track width and distance from center as input parameter. Learn more about reward function examples here



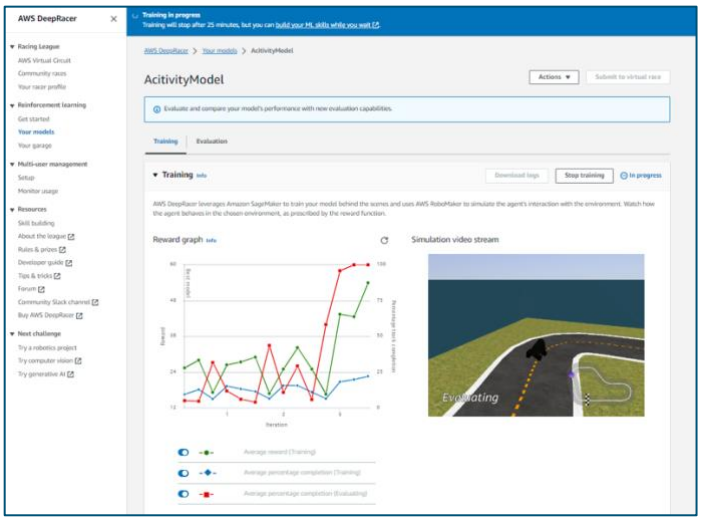
20. Under **Stop conditions**, select **25** minutes
21. Under Automatically submit to the DeepRacer race, uncheck the option **Submit this model to the following race after training completion**.
22. Choose **Create model** to start creating the model and provisioning the training job instance.
    Note: The initialization process takes a few minutes to change from **Initializing** to **In progress**.
23. Inspect training after the training instance is provisioned and the training job is initialized. Watch training in progress to see how your agent attempts to complete the track in the simulator.

Task 2: Interpret the Reward graph

The Reward graph shows three lines: green line, blue line and red line.

- **Green line** shows the average score your model has earned through the reward_function. It should increase over time as your model converges, which means that the reward_function you created makes the car finish the lap.
- **Blue line** shows the percentage of lap completion during the training, and it's normal for it to be below 100% because the model is in "exploration mode" during training, which means it's trying new actions to learn what the outcome is.
- **Red line** represents the average lap completion during the evaluation.

The following image shows a stable model as the red line is at 100% for a couple data points. The green line is also growing, which means we have a solid model.

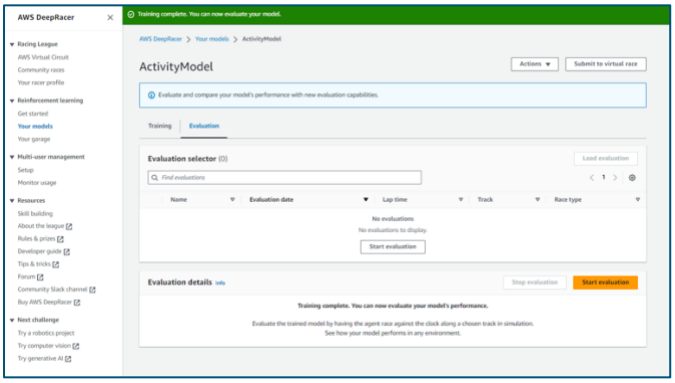


24. Interpret you reward graph. Use your text editor to write your graph interpretation.

Task 3: Evaluate your model

After the training is complete, you can evaluate your model. During evaluation, the model will select the best action based on the current input.

25. Choose the **Evaluation** tab.
26. Choose **Start evaluation**.

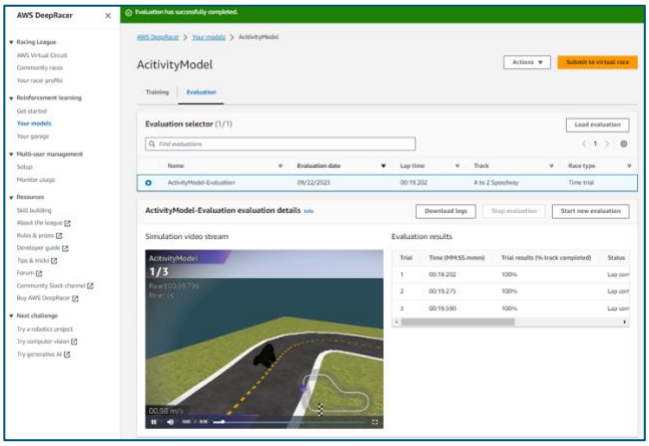


27. Under **Race type**, add a name for the evaluation you are performing on your model. Example, ActivityModelEvaluation.
28. choose **Time trial.**
29. Under **Evaluate criteria**, choose **A to Z Speedway.**
30. Select **Clockwise** as the track direction.
31. Under **Virtual race submission**, uncheck the option **Submit this model to the following race after training completion**.
32. Choose **Start evaluation**.
33. Check your simulation and Evaluation results.

Task 4: Interpret the Evaluation results

The evaluation will load the model into the 3D simulated environment and test how it performs. Once the evaluation is complete, you will see the results table. There is a slider in the AWS Console to access more fields than those visible in the following screenshot. One of the most important fields is the the Off-Track column.

The Off-track column is the number of times the car went off-track during the simulation. As we can see in this example, the model we're evaluating didn't go off-track during the evaluation, and the lap times are all pretty close, which means that the model is stable. Learn more about Train and evaluate models here



34. Interpret you the evaluation results. Use your text editor to write your interpretation.

Task 5: Develop a strategy to enhance the model

When training your model, there are a several strategies you can follow. Tweaking the reward function to include more parameters or changing how you reward the model can be one. You can also change the Action Items in the Action Space, for example to increase the speed of your model if it's stable but not fast enough. You can also fine-tune the hyperparameters if you feel adventurous. However, you need to be systematic and change only one thing at a time to be able to determine if the change you made improved your model. If you make multiple changes, you may not be able to tell which changes were good and which were not.

Changing your reward function will usually have an effect on your car's behavior. You can introduce some parameters that incentivize speed, or try to get the optimal trajectory using waypoints. Once your reward function creates a solid model, you can try to make it faster by increasing the speed in the action items, that help the model to gain experience driving faster.

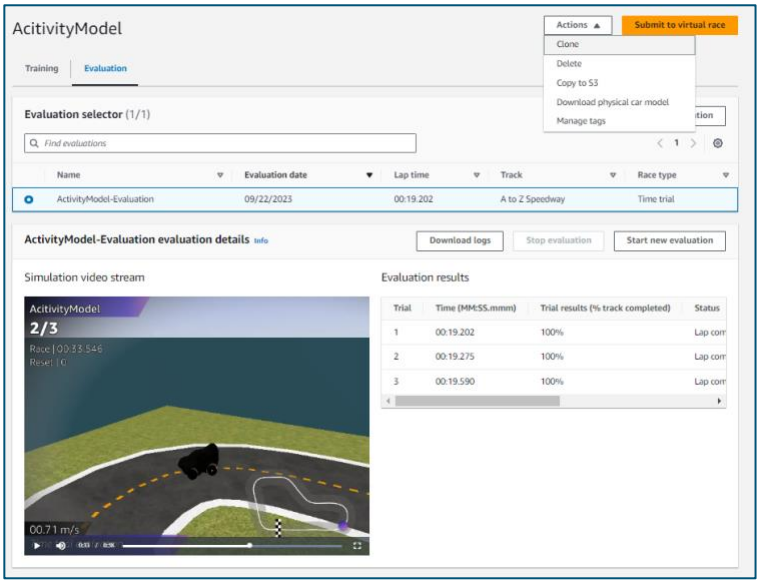
35. Hyperparameters affect the way the model learns. How should it weight new experience versus old knowledge? How much exploration versus exploitation? Use your text editor to record your answers.

Task 6: Enhance your model and reward function

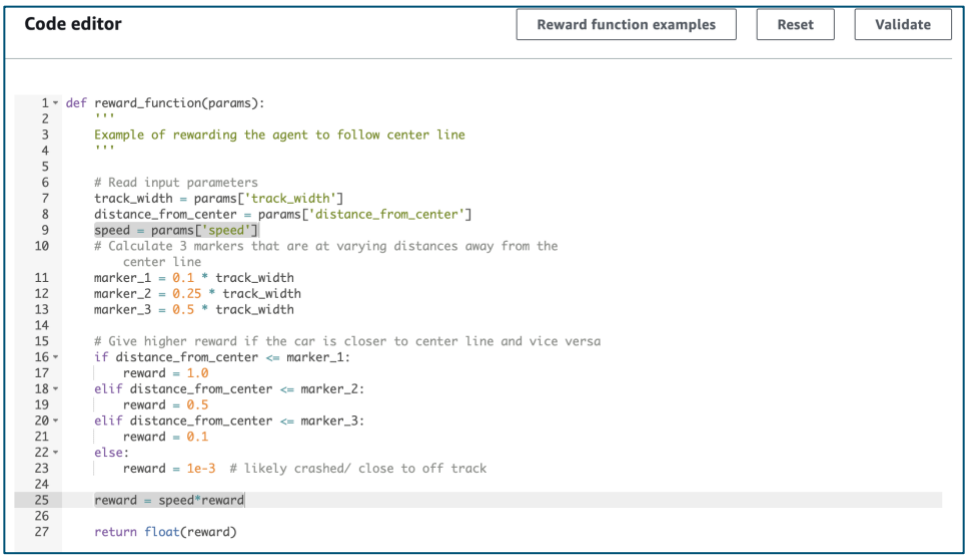
To take advantage of the prior training, you can start the new training by cloning the previously trained model, passing along the previously learned knowledge. You can follow this pattern to gradually add more features to the reward function to train your AWS DeepRacer vehicle.

In this task you will change only one perparameter at a time (Reward function, Action items, Hyperparameters, or Circuit), to understand the effects on the model. If you make several changes, it will be hard to understand which one was successful.

36. In the upper-right corner of the page, click on Actions to open the drop-down menu. Choose **Clone**.



37. Under **Training details**, add a name for the new model. Example, ActivityModelClone.
38. Under **Environment simulation**, choose **A to Z Speedway.**
39. Select **Clockwise** as the track direction.
40. Choose **Next**.
41. Under **Race type**, choose **Time trial.**
42. Choose **Next**.
43. Don't change any parameter under **Define discrete action space.**
44. Choose **Next**.
45. Under **Vehicle shell with sensor configuration**, choose **The Original DeepRacer** vehicle.
46. Choose **Next**.
47. Under the **Reward function** section, modify the sample code to enhance your reward function. Here you need to multiply the reward by the **speed**, thus higher speeds will lead to more reward points and ultimately faster laps.
    - After line 8 add **speed = params['speed']**
    - After line 23 insert a space and add **reward = speed*reward**



48. Under **Stop conditions,** set **25** as the Maximum time
49. Under **Automatically submit to the DeepRacer race**, uncheck the option **Submit this model to the following race after training completion**.
50. Choose **Create model**.
    Note: The initialization process takes a few minutes to change from **Initializing** to **In progress**.
51. Watch the **Reward graph** and **Simulation video stream** to observe the progress of your training job. You can choose the refresh button next to **Reward graph** periodically to refresh the **Reward graph** until the training job is complete.
52. Compare the new Reward graph with the Reward graph obtained in Task 2. Use your text editor to record your analysis.
53. Evaluate your new model.
54. Compare the two models. Decide which model is better and why. Use your text editor to record your results and analysis.

Task 7: Explore additional AWS DeepRacer learning resources

Explore the additional resources listed below to learn more about AWS DeepRacer. They include AWS documentation, community blogs, and training courses. Review the following recommended links:

- AWS DeepRacer Product Page
- AWS DeepRacer Student
- AWS DeepRacer for Educators

Activity checklist: Reinforcement Learning with AWS DeepRacer

The following checklist can be used to verify tasks competition and can be used by educators and learners.

Task	Accomplished	Partially Accomplished	Not Accomplished
Task 1: Create a model to get started	☐	☐	☐
Task 2: Interpret the Reward graph	☐	☐	☐
Task 3: Evaluate your model	☐	☐	☐
Task 4: Interpret the Evaluation results	☐	☐	☐
Task 5: Develop a strategy to enhance the model	☐	☐	☐
Task 6: Enhance your model and reward function	☐	☐	☐
Task 7: Explore additional AWS DeepRacer learning resources	☐	☐	☐