AWS DEEPRACER STUDENT EDUCATOR PLAYBOOK (CURRICULUM)

OVERVIEW

This playbook provides all the information and resources necessary for educators to implement AWS DeepRacer Machine Learning curriculum in the classroom through AWS DeepRacer Student modules. The curriculum outlines each module's overview, learning objectives, learning outcomes, key concepts, support material, and assessment and activity suggestions.

TABLE OF CONTENTS

MODULE 1.1 - MACHINE LEARNING REFRESHER

Overview

Students are provided with a high-level summary of how machine learning fits into the broader field of artificial intelligence, along with exploring the three core different types of machine learning: supervised, unsupervised, and reinforcement learning - which then leads well into the remainder of this course which focuses on the application of reinforcement learning in AWS DeepRacer.

This module encourages the student to consider whether machine learning is appropriate to address particular problems which they may face. If so, which type of machine learning is the most suitable to address the problem.

Learning Objectives

Students will develop:

- An understanding of the distinction and interrelationship between the field of artificial intelligence and domain of machine learning
- Knowledge and understanding of the three types of machine learning
- Ethics

Learning Outcomes

A student:

- Defines the terms artificial intelligence and machine learning
- Describes the relationship between artificial intelligence and machine learning
- Describe the three types of machine learning
- Identifies appropriate use-cases for each type of machine learning

Key Concepts

- Artificial intelligence
- Machine learning
- Supervised learning
- Unsupervised learning
- Reinforcement learning

Support Material

- Chapter 1.1 - Machine Learning Refresher

Assessment and Activity Suggestions

- **Interactive discussion:** this could be used at the beginning of the module to discuss the concept of machine learning, its applications, and impact on various industries. This would also be a useful calibration tool to see how much the students already understand about machine learning.
- **Case studies:** present real-world case studies where machine learning has been used successfully - with particular focus on the problem being addressed, the data being used, the machine learning type (and possibly algorithms, for a deeper-dive), and outcomes achieved.
- **Ethical considerations:** while not explicitly covered in this course, it could be insightful to facilitate a discussion on the ethical implications of machine learning - such as where machine learning has had negative consequences perhaps due to unfairness or bias, which can lead onto a discussion about fairness in machine learning.
- **Quiz:** match a use-case to the most appropriate type of machine learning.
- **In-class activity:** students are presented with a variety of problems and they need to explain why or why not the problem is suitable for machine learning (supervised, unsupervised, and reinforcement learning.
- **In-class activity:** students use their imagination to devise problems which could be solved with machine learning and then categories them into supervised, unsupervised, or reinforcement learning problems.
- **Research task:** AWS DeepRacer uses deep learning to take images (from the on-board camera), run these through an artificial neural network, and then map those to actions (accelerate, break, steer). Research how artificial neural networks work, particularly those used for image recognition. For bonus marks, reference how supervised and unsupervised learning can be used when training artificial neural networks.
- **Interactive discussion:** Why would we use deep learning algorithms instead of traditional machine learning algorithms? Conversely, why would we ever use traditional machine learning when we have deep learning? What are some use cases for each approach.

MODULE 1.2 - INTRODUCTION TO REINFORCEMENT LEARNING

Overview

Students are introduced to the reinforcement learning type of machine learning. This module provides students with a holistic understanding of reinforcement learning through looking its individual components but also how they work together in order to train and produce a machine learning model.

Three examples are also provided for use-cases of this type of machine learning and how the components of reinforcement learning work together in these cases to meet the requirements.

Learning Objectives

Students will develop:

- Knowledge and understanding about how reinforcement learning can be utilized to train a machine learning model

Learning Outcomes

A student:

- Describes how reinforcement learning works at a conceptual/high level
- Identifies and defines the four key concepts of reinforcement learning
- Identifies appropriate use-cases for reinforcement learning and how the key concepts of reinforcement learning can be applied

Key Concepts

- Reinforcement learning
- Agent
- Environment
- Actions
- Rewards

Support Material

- Chapter 1.2 - Introduction to reinforcement learning

Assessment and Activity Suggestions

- **In-class activity:** students use their imagination to devise problems which could be solved with reinforcement learning

- **Links to Generative AI:** reinforcement learning can also be used as part of Generative AI training, through a process called Reinforcement Learning Human Feedback (RLHF). In fact, ChatGPT was trained using this approach. Conduct research into how RLHF can be used to train a Generative AI model and the benefits it can provide.

MODULE 1.3 - REINFORCEMENT LEARNING WITH A MECHANICAL COMPUTER

Overview

Reinforcement learning is a concept, not purely a technology. To emphasize the conceptual aspects of reinforcement learning this module demonstrates how a simple mechanical computer that uses 24 matchboxes filled with beads can use reinforcement learning to learn and master Hexapawn, a mini-chess game.

Learning Objectives

Students will develop:

- Knowledge and understanding of how reinforcement learning works at a deeper conceptual level using a game with limited complexity

Learning Outcomes

A student:

- Identifies how the key components of reinforcement learning translate to real-world analogues in the game of Hexapawn
- Explains the gameplay of Hexapawn
- Describes how the matchboxes in Hexapawn collectively "learn" based on past experience of winning or losing the game

Key Concepts

- Game board
- Game pieces
- Piece movement
- Win condition
- Decision making
- Dynamic learning

Support Material

- Chapter 1.3 - Reinforcement learning with a mechanical computer
- "How to build a game-learning machine and then teach it to play and win" by Martin Gardner, Scientific American: http://cs.williams.edu/~freund/cs136-073/GardnerHexapawn.pdf

Assessment and Activity Suggestions

- **Player vs. Player Hexapawn:** two students play each other at Hexapawn, each recording how many wins and losses they incur. This will provide the students with an opportunity to familiarize themselves with the game and also get a sense for their success rate when versing another human
- **Player vs. Computer Hexapawn**: at this next step the matchboxes are introduced to simulate the "computer" and the student now versus the computer, once again recording how many wins and losses they incur. Compare this to their win/loss rate when versing another player.

MODULE 2.1 - REINFORCEMENT LEARNING WITH AWS DEEPRACER

Overview

There are a number of different applications of reinforcement learning and this topic introduces students to how reinforcement learning is used in AWS DeepRacer. The fundamental concepts and components of reinforcement learning are specifically applied and tied back to AWS DeepRacer and how the AWS DeepRacer can be trained to autonomously drive around a track.

Learning Objectives

Students will develop:

- An understanding of the AWS DeepRacer product and service
- Knowledge and understanding of how the key components of reinforcement learning apply to AWS DeepRacer

Learning Outcomes

A student:

- Defines what is the AWS DeepRacer along with describing its function and purpose
- Explains, at a conceptual level, how reinforcement learning is used by AWS DeepRacer
- Describes the key components of reinforcement learning and how they apply to AWS DeepRacer

Key Concepts

- Agent (AWS DeepRacer vehicle)
- Environment (track)
- State (camera images)
- Actions (accelerating, breaking, turning)
- Rewards (feedback from the environment in the form of a numerical value)
- Episode (cycle of the agent performing an action in the environment and receiving a reward)

Support Material

- Chapter 2.1 - Reinforcement learning with AWS DeepRacer
- AWS Documentation: What is AWS DeepRacer?
- AWS Documentation: AWS DeepRacer concepts and terminology
- AWS Documentation: Reinforcement learning in AWS DeepRacer

Assessment and Activity Suggestions

- **Interactive animation:** students create an interactive animation showing an episode of training along with the interaction of all the components in reinforcement learning, similar to that shown in the instructional video for this module. The students should also provide an explanation of what the components are doing with specific reference back to their application in AWS DeepRacer.
- **Rule-based racing:** if an AWS DeepRacer vehicle is available then students could race the AWS DeepRacer using manual control, following a set of pre-defined rules for the behavior of the vehicle. This allows the student to understand the challenges associated with the decision-making process when engaging in autonomous racing.
- **Community engagement:** students can join the AWS DeepRacer Student Discord where they can collaborate with other racers, ask questions, and get tips through the student AI/ML Discord channel: https://discord.com/invite/G72rNQmJRg

MODULE 2.2 - TRAINING YOUR FIRST AWS DEEPRACER MODEL

Overview

The AWS DeepRacer console removes much of the friction associated with traditionally training machine learning models, abstracting away the complexities and instead presenting the user with a unified and simplified interface to training their reinforcement learning models.

This module provides students with a step-by-step walk-through of training an AWS DeepRacer model using the AWS DeepRacer console. The purpose of this module is to provide an overview of the console, rather than a deep dive into the various training parameters - which is instead covered in future modules.

Learning Objectives

Students will develop:

- Skills in training a reinforcement learning model for AWS DeepRacer through the AWS DeepRacer console
- Understanding, at a high-level, of the various parameters which can be specified when training an AWS DeepRacer model

Learning Outcomes

A student:

- Uses the AWS DeepRacer console to train a reinforcement learning model for AWS DeepRacer
- Identifies the different parameters which can be specified when training an AWS DeepRacer model
- Explains the interrelationship between the steps in the AWS DeepRacer console for training a model

Key Concepts

- AWS DeepRacer track
- Reinforcement learning training algorithms: PPO (Proximal Policy Optimization) and SAC (Soft Actor Critic)
- Reward function
- Training time vs. model quality

Support Material

- 2.2 - Training your first AWS DeepRacer model
- AWS Documentation: Train your first AWS DeepRacer model (note that some options, particularly hyperparameters and action space type selection are not available in AWS DeepRacer Student)
- AWS Documentation: AWS DeepRacer training algorithms

Assessment and Activity Suggestions

- **Community (private) race:** if the course instructor has a full AWS account they can setup a Community Race allowing you to run your own, private, invitation-only virtual race. The instructor sets up the race in their AWS account and can then invite the students to participate. This could be used for some informal, fun racing amongst the class or a more serious race. Community races for AWS DeepRacer Student currently does not offer live virtual racing. Educators can use multi user accounts to allow for live virtual racing explained further in the lab's playbook. Advantages of a non-live race are students can submit simultaneously to a single race, then watch evaluation replays on video for themselves and others in the classroom. The Community Races feature is compatible with AWS DeepRacer Student and instructions on setting-up these races are available here: https://docs.aws.amazon.com/deepracer/latest/developerguide/educator.html
- **AWS DeepRacer strategy analysis:** class discussion surrounding the different strategies which could be used to train AWS DeepRacer model, through the selection of training algorithm and tuning of the reward function. The modules which follow dive deeper into these aspects, so encouraging an early discussion at this stage could set the scene for what is to come.

MODULE 3.1 - TRAINING ALGORITHMS

Overview

There are many different algorithms which can be used for machine learning training, and it can be difficult to identify the features and differences between these algorithms to select the most appropriate for the given use-case. AWS DeepRacer supports two algorithms: Proximal Policy Optimization (PPO) and Soft Actor Critic (SAC).

This module provides a high-level, no math, overview of these two algorithms and how they can be used to train an AWS DeepRacer reinforcement learning model. This enables the student to make a more informed decision about which algorithm to use when training their AWS DeepRacer model.

Learning Objectives

Students will develop:

- Knowledge and understanding about how machine learning algorithms work, along with differences in these algorithms

Learning Outcomes

A student:

- Identifies the two training algorithms supported by AWS DeepRacer
- Describes the purpose of training algorithms in training an AWS DeepRacer model
- Describe the key components of machine learning algorithms and their purpose
- Compare and contrast between the PPO and SAC algorithms supported by AWS DeepRacer, at a high-level

Key Concepts

- Proximal Policy Optimization (PPO)
- Soft Actor Critic (SAC)
- Deterministic policy
- Stochastic policy
- Value function
- Policy update
- “On-policy” learning
- “Off-policy” learning

Support Material

- 3.1 - Training algorithms
- AWS Documentation: AWS DeepRacer training algorithms

Assessment and Activity Suggestions

- **Algorithm comparison:** Divide students into groups and assign each group one of the training algorithms (PPO or SAC). Each group then completes a deep dive into that algorithm, conducting research to further understand the key concepts, advantages, and limitations of the algorithm at a more detailed level than presented in this module. Each group can then present their findings to the class in the form of a presentation, encouraging a class discussion about the differences between the two algorithms.
- **Race competition:** Setup a private Community Race (see Module 2.2) and each student either individually, or in small groups, trains a model using either PPO or SAC. The models are then raced and the students share the strategies they implemented for training, providing an opportunity for the students to apply their knowledge of training algorithms and witness how the models perform in a physical race.

MODULE 3.2 - REWARD FUNCTIONS

Overview

Developing a reward function is one of the most important jobs of an AWS DeepRacer developer. In this module, students will have the opportunity to deep dive into the details of reward function development and gain a greater understanding of how the reward function can be used to differentiate their AWS DeepRacer model from others.

Learning Objectives

Students will develop:

- Skills in designing and developing reward functions for AWS DeepRacer
- Understanding that the development of the reward function is an iterative process

Learning Outcomes

A student:

- Describes the purpose of the reward function when training an AWS DeepRacer model
- Explains how the AWS DeepRacer sample reward functions work
- Describes the most common input parameters which can be used in the reward function
- Uses a variety of algorithm development approaches to write the reward function
- Describes the likely effect of modifying a reward function will have on the model which results from training

Key Concepts

- Reward function
- Python programming language
- Input parameters

Support Material

- 3.2 - Reward functions
- AWS Documentation: AWS DeepRacer reward function examples
- AWS Documentation: Input parameters of the AWS DeepRacer reward function

Assessment and Activity Suggestions

- **Real-world analogies:** Ask the students to devise real-world situations where a reward function could be used. This does not need to be a situation involving coded reward functions, it could be just a

simple written procedure. For example, training a dog - if the dog does the correct thing it receives a treat, if it does not then it receives no treat.

- **Analyzing existing reward functions:** Provide the students with an existing pre-built reward function. Ask them to analyze and discuss the effectiveness of the reward function in encouraging the desired behavior, and suggest possible improvements.
- **Designing simple reward functions:** Divide students into small groups and assign each group to design a reward function which incentivizes the AWS DeepRacer model to complete a particular task efficiently (for example, get around the track as fast as possible, sacrificing accuracy; or get around the track as accurately as possible, sacrificing speed). Once the reward functions are developed, train a model for 1 - 2 hours and then run it in a community race to allow all the groups to compare their different models and efficacy of their reward functions.
- **Parameter tuning:** Provide the students with a baseline reward function and ask them to experiment with changing some of the parameters used in order to optimize and improve the performance of the reward function.
- **Links to Generative AI:** This module provided students with experience fine-tuning their reward functions. Fine-tuning is an important skill, particularly in the space of Generative AI and using Foundation Models (FM), generalized pretrained models which can be fine-tuned for particular use-cases. Research the use of Foundation Models and how they can be fine-tuned. Some topics which students may wish to consider include:

a. **Instruction-based fine tuning (prompt engineering):** zero-shot learning, few-shot learning

b. Domain adaption fine-tuning

MODULE 3.3 - HOW THE AWS DEEPRACER DEVICE WORKS

Overview

While the AWS DeepRacer console abstracts away much of the complexity involved in training an AWS DeepRacer model, the AWS DeepRacer itself is anything but simple - it is a complex combination of technologies which allows the DeepRacer to take input images at a rate of 15 times per second, analyze these images, and make a decision in real-time all while the vehicle is driving around the track.

This module takes a deep drive “under the hood” of the AWS DeepRacer to learn how the vehicle makes decisions in real-time, the challenges involved, and how these challenges are addressed on the AWS DeepRacer through the process of model optimization.

Learning Objectives

Students will develop:

- Knowledge and understanding about how the AWS DeepRacer performs real-time inference at edge
- An understanding of the challenges in real-world applications of machine learning inference and how these challenges can be addressed

Learning Outcomes

A student:

- Defines *machine learning inference* and its purpose in the machine learning process
- Identify factors which can affect the inference rate and inference time/latency
- Describe the challenges of machine learning real-time inference at edge and how these challenges can be addressed
- Explain how the Intel OpenVINO toolkit is used in AWS DeepRacer to assist with real-time inference

Key Concepts

- Machine learning inference
- Inference rate
- Inference time/latency
- Real-time inference at edge
- Inference engine
- Intel OpenVINO toolkit
- Model optimization process

Support Material

- 3.3 - How the AWS DeepRacer device works
- GitHub: AWS DeepRacer
- AWS DeepRacer Projects

Assessment and Activity Suggestions

- **Extending AWS DeepRacer:** The AWS DeepRacer's device software has been open-sourced on GitHub (https://github.com/aws-deepracer/) and this provides significant opportunities for end-user extension of the AWS DeepRacer device. A challenge activity for students could be to implement a custom behavior on AWS DeepRacer, such as those shown on the AWS DeepRacer Projects page.

MODULE 3.4 - PRO TIPS FROM THE PROS

Overview

Community is a key aspect of AWS DeepRacer, being able to discuss and collaborate with other developers from around the world with the view to improving your own AWS DeepRacer model. In this module, students will have the opportunity to hear from a variety of pro-level racers who are providing some of their tips and tricks about how to succeed in AWS DeepRacer.

Learning Objectives

Students will develop:

- Knowledge and understanding of how pro-level racers in the AWS DeepRacer League train their models

Learning Outcomes

A student:

- Describes additional new ways in which they can train and tune their AWS DeepRacer models

Support Material

- 3.4 - Pro tips from the pros

Assessment and Activity Suggestions

- **Community engagement:** students can join the AWS DeepRacer Community Slack where they can collaborate with other racers, ask questions, and get tips through the student AI/ML Discord channel: https://discord.com/invite/G72rNQmJRq