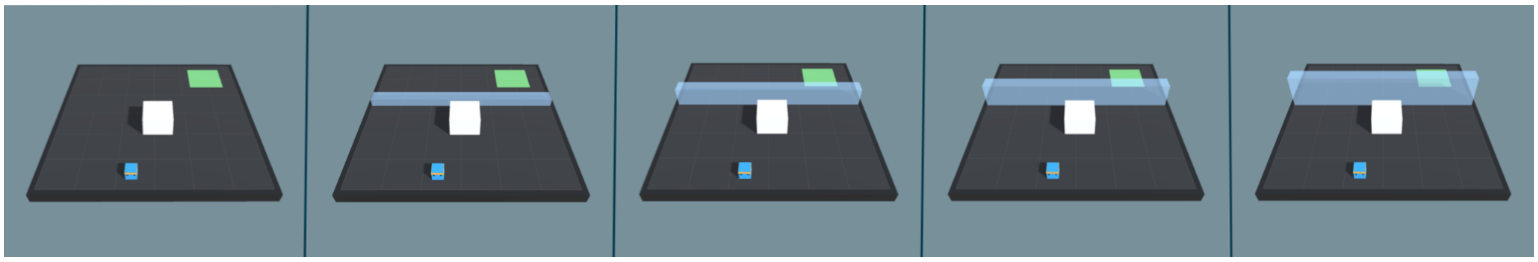
Training Agents with Curriculum Learning

Solving Complex Tasks using Curriculum Learning

Curriculum learning is a way of training a machine learning model where more difficult aspects of a problem are gradually introduced in such a way that the model is always optimally challenged. This idea has been around for a long time, and it is how we humans typically learn. If you imagine any childhood primary school education, there is an ordering of classes and topics. Arithmetic is taught before algebra, for example. Likewise, algebra is taught before calculus. The skills and knowledge learned in the earlier subjects provide a scaffolding for later lessons. The same principle can be applied to machine learning, where training on easier tasks can provide a scaffolding for harder tasks in the future.

Imagine training the medic to scale a wall to arrive at a wounded team member. The starting point when training a medic to accomplish this task will be a random policy. That starting policy will have the medic running in circles, and will likely never, or very rarely, scale the wall properly to revive their team member (and achieve the reward). If we start with a simpler task, such as moving toward an unobstructed team member, then the medic can easily learn to accomplish the task. From there, we can slowly add to the difficulty of the task by increasing the size of the wall until the medic can complete the initially near-impossible task of scaling the wall. We have included an environment to demonstrate this with ML-Agents, called Wall Jump.

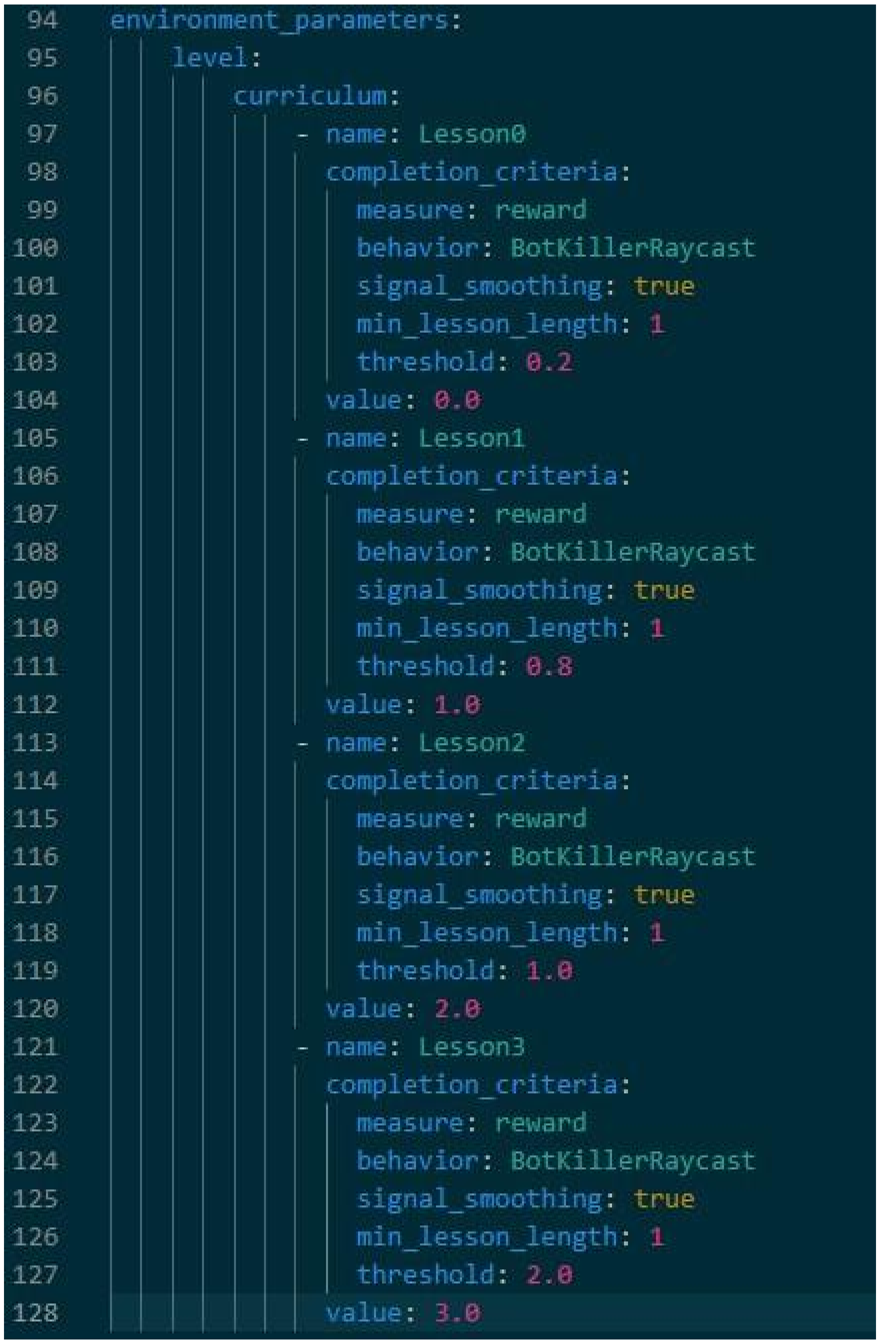


Unity Wall Jump Git Repo:
https://github.com/Unity-Technologies/ml-agents/tree/main/Project/Assets/ML-Agents/Examples/WallJump

Article: https://github.com/Unity-Technologies/ml-agents/blob/main/docs/ML-Agents-Overview.md#solving-complex-tasks-using-curriculum-learning

Creating the Curriculum

To train an agent using curriculum learning you will need to add environment parameters to your .yaml file. The “curriculumLearning.yaml” file example is already set up with environment parameters you can look at.



```yaml
environment_parameters:
    level:
        curriculum:
            - name: Lesson0
              completion_criteria:
                measure: reward
                behavior: BotKillerRaycast
                signal_smoothing: true
                min_lesson_length: 1
                threshold: 0.2
              value: 0.0
            - name: Lesson1
              completion_criteria:
                measure: reward
                behavior: BotKillerRaycast
                signal_smoothing: true
                min_lesson_length: 1
                threshold: 0.8
              value: 1.0
            - name: Lesson2
              completion_criteria:
                measure: reward
                behavior: BotKillerRaycast
                signal_smoothing: true
                min_lesson_length: 1
                threshold: 1.0
              value: 2.0
            - name: Lesson3
              completion_criteria:
                measure: reward
                behavior: BotKillerRaycast
                signal_smoothing: true
                min_lesson_length: 1
                threshold: 2.0
              value: 3.0
```

The environment parameter created here is called “level”.
The value of “level” changes with each lesson your agent goes through during training.
At “Lesson0”, the value of “level” is 0.0.
At “Lesson1”, the value of “level” is 1.0.
At “Lesson2”, the value of “level” is 2.0.
At “Lesson3”, the value of “level” is 3.0.

Each lesson has a threshold reward value your agent needs to reach before moving onto the next lesson.
For “Lesson0”, the threshold reward value is 0.2.
For “Lesson1”, the threshold reward value is 0.8.
For “Lesson2”, the threshold reward value is 1.0.
For “Lesson3”, the threshold reward value is 2.0.
Challengers should increase the reward threshold if they want their agent to be more proficient at the specified level before moving on.

Challengers should also decrease the reward threshold if they want their agent to move on to the next level more quickly in training.

The “min_lesson_length” value specifies how many times your agent will need to reach the reward threshold before moving onto the next lesson.
For all lessons, the min_lesson_length is 1.

How to Move onto the Next Level

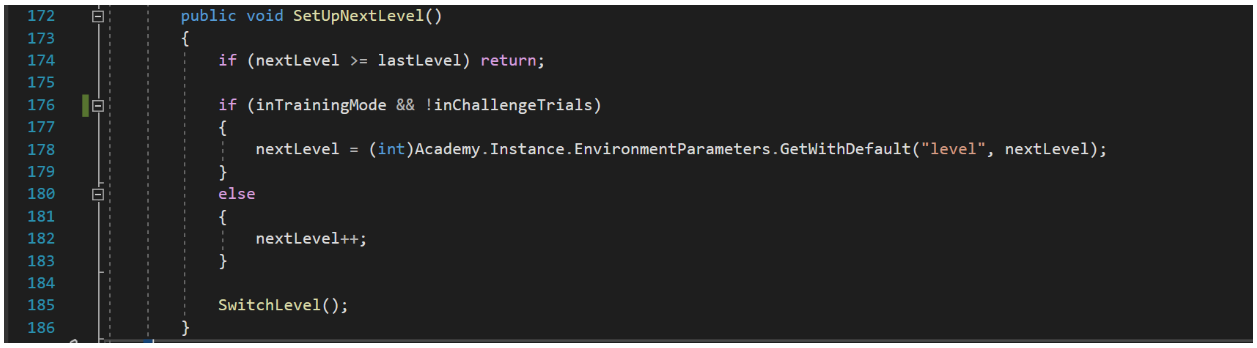
Level advancement in the Training Trials is dependent on 2 criterias:

1. All enemy bots have been defeated
2. The reward threshold has been met

Level advancement in the Challenge Trials is dependent on 1 criteria:

1. All enemy bots have been defeated

How The Level Environment Parameter Interacts with the Code



```
public void SetUpNextLevel()
{
    if (nextLevel >= lastLevel) return;

    if (inTrainingMode && !inChallengeTrials)
    {
        nextLevel = (int)Academy.Instance.EnvironmentParameters.GetWithDefault("level", nextLevel);
    }
    else
    {
        nextLevel++;
    }

    SwitchLevel();
}
```

Line 178 sets the nextLevel variable based on the “level” parameter found in the .yaml file's Environment Parameters.

Works Cited

Unity-Technologies. “Ml-Agents/ML-Agents-Overview.md at Main · Unity-Technologies/Ml-Agents.” GitHub, 16 Apr. 2021, github.com/Unity-Technologies/ml-agents/blob/main/docs/ML-Agents-Overview.md#solving-complex-tasks-using-curriculum-learning.