CSCI 599 Technical Paper for Final Project

DAMI
Ying Cheng, Chien-Hao Huang, Haichao Ma, Ta-Yang Wang, Lijie Zhao

University of Southern California

1 Introduction

Deep learning is a class of AI calculations that utilizes various layers to dynamically separate more significant level features from the crude information [1]. When prepared appropriately, a deep learning model gets ready to perform a huge number of normal, dull assignments inside a generally shorter time-frame contrasted with what it would take for a person.

Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment to maximize the notion of cumulative reward [2]. Without an adequate data-set, it will undoubtedly gain from its experience. It is additionally planned to accomplish the perfect conduct of a model inside a particular setting, to boost its performance, which is ideal in regards to the goal we aim to achieve.

The advantage of the high-quality physics, graphics, and simple developer control provided by the Unity Engine and Editor were taken by the team to fulfill our needs. The ML-Agent [3], Unity Machine Learning Agents Toolkit, allows us to transform games and simulations created using the Unity Editor into environments where intelligent flying sledges in our project can be trained to learn how to fly along the track without bumping into any obstacles.

1.1 Background

A racing game in a snowy terrain is built in Unity, flying sledges as agents should go through the race track without crashing into obstacles. Once the sledges coinciding with obstacles, they will then crash immediately. Specifically, agents should fly through each checkpoint within few seconds and complete all two laps of the race. we design a corresponding reward function to make agents accomplish these tasks while racing.

2 Related Work

There are various reinforcement learning algorithms that can be used for games [4, 5], such as state action reward state action (SARSA) [6], the deep Q-network (DQN) [7], deep deterministic policy gradient (DDPG) [8], trust region policy optimization (TRPO) [9], and proximal policy optimization (PPO) [10]. Although each algorithm shares its strengths and weaknesses, we need to find the one that best fits our game. Both SARSA and DQN can only work in discrete action decisions, so they fail to provide continuous control in our racing game. DDPG uses raw Q values to make the decision, which learns much slower than advantage learning since the state of our game slightly changes on each time step. Since PPO modifies the surrogate objective function from TRPO and thus improves the performance, we decide to apply PPO-based algorithm to our game.

3 Methodology

3.1 Proximal Policy Optimization

We make use of ML-Agent, which provides an implementation of PPO algorithm. PPO computes advantage estimates, that is, Q values are compared against other values for a variety of actions to determine which action is better. We then train our sledge agent to travel around the race track. The main idea is that we want to reward the agent if it flies through each checkpoint and punish the agent if we run out of time or crashes into obstacles.

3.2 Curriculum learning

The strategy of training machine learning models on a series of progressively harder problems is known as curriculum learning [11]. Introducing tasks with increasing order of difficulty could potentially speed up online training. Our goal here is to make our sledge agent learn how to pass through checkpoints. We quantify the difficulty of a task based on the radius of checkpoints, the larger the easier. Therefore, we first start with a much larger radius, so that the sledge can easily get a positive reward. Then we gradually shrink the size of the checkpoint radius.

4 Experimental Results

We utilize the hyperparameters in the build-in learning environment **pyramids** in the Unity ML-Agents with some modification. The reason is that it moves around the environment and uses raycasts to check for obstructions, which perfectly fit our sledge racing game. We train the PPO model with the following hyperparameters: batch size 512, buffer size 4096, hidden units 128, the number of layers is 2, and the number of epoches is 6.

We measure the success based on the returned cumulative reward for the curriculum learning. We define the reward function with (1) a small negative reward $-1/MaxStep$ for each time step, where $MaxStep = 5000$, (2) a negative reward -0.5 for time out, (3) a negative reward -1 for crushing into obstacles, and (4) a positive reward $+0.5$ for successfully passing through a checkpoint. The thresholds of the curriculum are set to be $1, 2, 4, 6$, by which the lessons will be increased. Each time the task becomes slightly more difficult and forces the agent to get further. The lesson will increment after the average cumulative reward of the last 100 episodes exceeds the current threshold. We start with a radius of 50, and then decrease it down to $30, 20, 10$, and 0.

Figure 1 shows the cumulative reward of the PPO model by curriculum learning for over one million steps. As can be seen in the figure, the algorithm eventually converges, which suggests that PPO algorithms have the ability to solve our sledge racing problem. Furthermore, the cumulative reward significantly increased in the first 100k steps, despite the structure of our racing game is complicated. It is highly possible that curriculum learning can generalize faster.

Figure 2 indicates the curiosity reward of the model. We can see that the model with adoption of the curriculum is efficient. Furthermore, the intrinsic motivation encourages certain agent actions and hence helps the agent achieve higher reward with much less steps. It demonstrates that the efficiency of training can be improved by providing a systematic instruction for the agent.

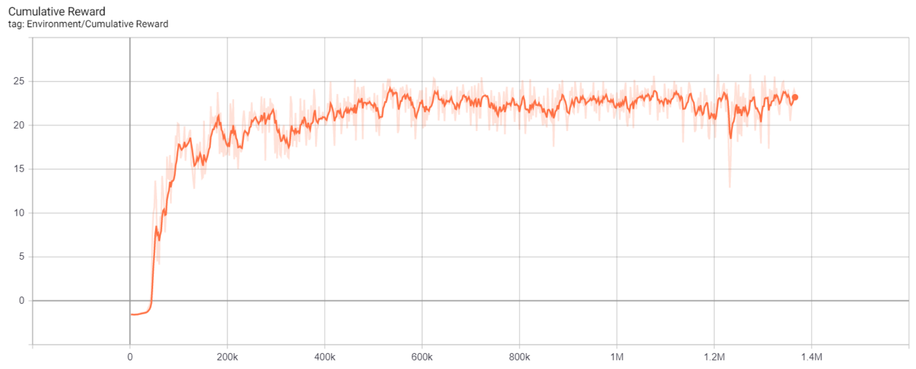

Fig. 1. The cumulative reward of the PPO model with curriculum in each step

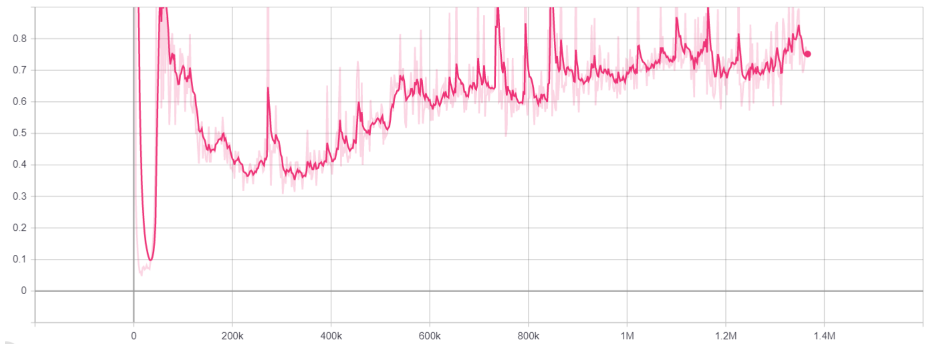

Fig. 2. The curiosity reward of the PPO model with curriculum in each step

5 Conclusion

We implemented the PPO algorithm with curriculum learning and designed a customized reward function in our sledge racing game. The result shows that by carefully tuning the training hyperparameters, the PPO models can be applied our customized racing game. Furthermore, designing a curriculum like gradually shrinking the radius of the area of positive rewards can speed up learning.

6 Future Work

We can try to implement other frameworks of proposing curriculum that fit our sledge racing game, such as teacher-student curriculum learning [12], asymmetric self-play [13], or automatic goal generation [14]. Besides curriculum learning, we can also try other possible learning methods and design a more detailed reward function to improve the training speed. In addition, we can also try to visualize the training process to help us understand when the neural network will get trapped into the local optimal solution.

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[2] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[3] Arthur Juliani et al. "Unity: A general platform for intelligent agents". In: *arXiv preprint arXiv:1809.02627* (2018).

[4] Kai Arulkumaran et al. "Deep reinforcement learning: A brief survey". In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 26–38.

[5] Yuxi Li. "Deep reinforcement learning: An overview". In: *arXiv preprint arXiv:1701.07274* (2017).

[6] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*. Vol. 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[7] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), pp. 529–533.

[8] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[9] John Schulman et al. "Trust region policy optimization". In: *International conference on machine learning*. 2015, pp. 1889–1897.

[10] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[11] Yoshua Bengio et al. "Curriculum learning". In: *Proceedings of the 26th annual international conference on machine learning*. 2009, pp. 41–48.

[12] Tambet Matiisen et al. "Teacher-student curriculum learning". In: *IEEE transactions on neural networks and learning systems* (2019).

[13] Sainbayar Sukhbaatar et al. “Intrinsic motivation and automatic curricula via asymmetric self-play”. In: *arXiv preprint arXiv:1703.05407* (2017).

[14] Carlos Florensa et al. “Automatic goal generation for reinforcement learning agents”. In: *arXiv preprint arXiv:1705.06366* (2017).