Taxicab: Duckietown with Soft Actor-Critic

Julian Freedberg*
Department of Computer Science
University of California, Irvine
Irvine, CA 92617
jfreedbe@uci.edu

Bryon Tjanaka*
Department of Computer Science
University of California, Irvine
Irvine, CA 92617
btjanaka@uci.edu

Carleton Zhao*
Department of Computer Science
University of California, Irvine
Irvine, CA 92617
carletoz@uci.edu

Abstract

We use Soft Actor-Critic to train a reinforcement learning agent for the Lane Following challenge of the Artificial Intelligence Driving Olympics (AI-DO). Specifically, we leverage the existing implementation of Soft Actor-Critic in the Spinning Up library as a starting point and modify it to perform on progressively more difficult versions of Lane Following.

Thus far, we have successfully trained the default SAC agent in the simulated Duckietown environment, though we achieved subpar results. As such, we have also simplified our environment to a straight track and modified our network with convolutional layers. After only 40,000 training steps, we found that this modified network was occasionally able to complete this simplified track. Future work will focus on completing an agent that functions in a more complex loop track.

1 Introduction

Deep Reinforcement Learning (RL) has achieved tremendous success in various challenging tasks (1), (2). One area that has proven challenging for deep RL is that of self-driving cars. To address this problem, Paull et al. have created Duckietown, an educational platform where small "duckiebots" travel around a miniature city known as a "duckietown" (3).

We propose to further the reputation of deep RL by exploring its feasibility in Duckietown. Specifically, we aim to see whether deep RL can succeed in the relatively simple Lane Following (LF) task, where the duckiebot traverses a continuous track in the absence of other duckiebots. We begin by using a standard implementation of Soft Actor-Critic (4) to train an agent in a simulated Duckietown environment. We then study the consequences of modifying various aspects of the algorithm, model, observations, reward, and environment.

Our project is still in progress, but if we are successful, we expect to see a virtual duckiebot succeed in the LF task. As it turns out, this is not a trivial task, and our initial attempts resulted in an agent that merely spun around in place, likely because it was caught in a local optimum.

In our experiments, we have explored both a simple straight track and a more complex curved one. We have also changed the environment variables to obtain a more consistent and straight track for

*Equal contribution. Order determined alphabetically.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

training. On the model side, we have experimented with modifying the actor network to include convolutional layers which increase the efficiency of image processing.

We plan to experiment with rewards which provide a harsh penalty for turning in place or performing other negative behaviors. Following the work of (5), we also aim to explore ensembles of SAC models where each model is initialized with different random values, and the models' output is aggregated by averaging or majority voting.

2 Background

2.1 Reinforcement Learning and Soft Actor-Critic

We adopt the standard MDP formulation, where the MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, the transition function $p(s'|s, a)$ is the probability of transitioning to state s' when taking action a in state s, and the reward function $r(s, a)$ gives the reward for taking action a in state s, and γ is the discount factor. In reinforcement learning, the goal of the agent in this MDP is to find a policy $\pi(a|s)$ that maximizes the discounted return over its entire lifetime.

Soft Actor-Critic (SAC) is a state-of-the-art off-policy reinforcement learning algorithm (4). SAC relies heavily on maximum-entropy reinforcement learning, where in addition to maximizing the expected return of a policy, one maximizes its expected entropy. Thus, SAC trains two Q-networks and a policy network and achieves excellent sample efficiency.

2.2 Duckietown

Duckietown is a platform for evaluating the effectiveness of deep learning and classical algorithms on autonomous driving tasks of increasing difficulty. (3). The AI Driving Olympics (AI-DO) is a set of challenges for robots in Duckietown. It features several tasks, the simplest of which is Lane Following, or driving on the right-hand side of the road for as long as possible. More advanced tasks involve driving while interacting with other vehicles, and with intersections. Gym-Duckietown (6) is a simulator commonly used for the Duckietown platform. It is built on top of OpenAI Gym.

2.3 Libraries

Spinning Up (7) is a collection of implementations of various reinforcement learning algorithms intended for educational use. As such, the implementations, while not designed for efficiency, are easy to read and modify. Most of the algorithms in Spinning Up have implementations for both TensorFlow and PyTorch. We choose to use the PyTorch implementation of Soft Actor-Critic.

3 Problem Statement

We address the problem of developing an agent that can be trained with SAC to complete the LF task in Duckietown. We will experiment with different models, environments, and rewards to accomplish this task.

Our environment is Gym-Duckietown (6). The action space in Gym-Duckietown consists of two continuous values in the range [-1,1], one for the left and one for the right wheel. Observations from Gym-Duckietown consist of a 3-channel 640x480 image of the world – this image can be provided with fisheye distortion to mimic the real robot camera. Gym-Duckietown provides a variety of built-in city maps; we wll use a simple straight track and loop track before progressing to more complex cities. Finally, it should be noted that Gym-Duckietown also provides an option for domain randomization to assist in simulation-to-real transfer. We have turned off both the fisheye distortion and the domain randomization to decrease the complexity of the LF task.

Each episode in our environment terminates after 1000 timesteps. During those timesteps, the agent receives reward for having a velocity that aligns with an invisible Bezier curve that follows the right side of the road in the Duckietown environment. We will experiment with reward terms that punish behavior such as spinning in place.

3.1 Evaluation

We evaluate our success in two ways. Quantitatively, we measure episodic return and look at how it varies as the agent encounters more timesteps. Qualitatively, we look at videos of the robot performing in simulation and see whether it is really following the lane.

3.2 Milestones

Our milestones are as follows:

1. **Baseline:** Train and evaluate a library implementation of an SAC agent on a complex city in the simulated LF task. (Completed)
2. **Simple:** Add convolutional or other layers to the model and train it to complete a simple straight track. (Complete)
3. **Goal:** Modify the reward signal and model as needed to create an agent that successfully completes the loop track.
4. **Moonshot:** Train the agent to follow lanes in the complex city, perhaps on a physical duckiebot.

4 Methods

Our algorithm and model, including hyperparameters, all began as the defaults from Spinning Up's PyTorch implementation of Soft Actor-Critic.

4.1 Iteration 1

Our initial SAC implementation was mostly "out of the box." For the policy network, this implementation used a multilayer perceptron (MLP), so we had to flatten all of our observations, which were color images. The MLP thus had 57600 nodes in its input layer (3 channels of 160x120), 2 hidden fully-connected layers of 128 nodes each, and an output layer of 2 nodes. The Q-networks had identical strucutre.

4.2 Iteration 2

In out next iteration, we modified the policy network by adding 2 convolutional layers before the MLP and made all of our observations grayscale images. The first convolutional layer had 6 channels and a 5x5 kernel. The second convolutional layer had 16 channels and another 5x5 kernel. After each convolutional layer all the outputs go through a 2x2 max pooling layer and a relu function. We also reduced the MLP to have only 1 hidden fully-connected layer of 128 nodes. We did not add convolutional layers to our Q-networks, but we did reduce the number of hidden layers to 1.

5 Experiments

In each experiment, training was performed using the Nvidia GeForce GPU on one of our members' personal computers.

5.1 Baseline

For the Baseline experiment, we train our first iteration of SAC in the `udem1` map in Gym-Duckietown. This is a relatively complex map, with cities and intersections in addition to a main track that has multiple turns.

5.2 Simple

For the Simple experiment, we train our second iteration of SAC in a much simpler straight track, with the hope that we will see the agent exhibit some form of intelligent motion.

6 Results

6.1 Baseline

After training for 200,000 timesteps, we found that the agent merely spun in circles.[2] This is not surprising, as we are not taking advantage of any of the properties that images have, and this environment is also very complex.

6.2 Simple

After training for 40,000 timesteps, we found that the agent was able to make progress on the track. [3] On occasion, it could even make it all the way to the end of the track.

Acknowledgments

This research was performed as part of the Winter 2020 offering of CS 175: Project in Artificial Intelligence taught by Roy Fox at the University of California, Irvine.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the instructor.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. A. Riedmiller, and D. Silver, "Emergence of locomotion behaviours in rich environments," *CoRR*, vol. abs/1707.02286, 2017.

[3] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y. F. Chen, C. Choi, J. Dusek, Y. Fang, D. Hoehener, S. Liu, M. Novitzky, I. F. Okuyama, J. Pazis, G. Rosman, V. Varricchio, H. Wang, D. Yershov, H. Zhao, M. Benjamin, C. Carr, M. Zuber, S. Karaman, E. Frazzoli, D. Del Vecchio, D. Rus, J. How, J. Leonard, and A. Censi, "Duckietown: An open, inexpensive and flexible platform for autonomy education and research," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1497–1504, May 2017.

[4] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, (Stockholmsmässan, Stockholm Sweden), pp. 1861–1870, PMLR, 10–15 Jul 2018.

[5] S. Faußer and F. Schwenker, "Neural network ensembles in reinforcement learning," *Neural Processing Letters*, vol. 41, pp. 55–69, Feb 2015.

[6] M. Chevalier-Boisvert, F. Golemo, Y. Cao, B. Mehta, and L. Paull, "Duckietown environments for openai gym." `https://github.com/duckietown/gym-duckietown`, 2018.

[7] J. Achiam, "Spinning Up in Deep Reinforcement Learning," 2018.

[8] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, G. Dulac-Arnold, J. Agapiou, J. Leibo, and A. Gruslys, "Deep q-learning from demonstrations," 2018.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[10] J. Zilly, J. Tani, B. Considine, B. Mehta, A. F. Daniele, M. Diaz, G. Bernasconi, C. Ruch, J. Hakenberg, F. Golemo, A. K. Bowser, M. R. Walter, R. Hristov, S. Mallya, E. Frazzoli, A. Censi, and L. Paull, "The ai driving olympics at neurips 2018," *arXiv preprint arXiv:1903.02503*, 2019.

[2] Video of an evaluation episode for the Baseline: https://youtu.be/dgNZG6V5d9A

[3] Video of an evaluation episode for the Simple: https://youtu.be/lu4tbP2URtg