Autonomous drifting using Deep Reinforcement Learning

Rahul Karanam
University of Maryland College Park,MD,USA
rkaranam@umd.edu

Abstract -
One of the main goals of self-driving algorithms is to carry out manoeuvres safely, even if an abnormality occurs. When the tires lose their grip on the road, one of these motion planning issues arises; an autonomous vehicle should be able to handle this condition.

Controlling a car in a drifting motion is more difficult than controlling it in (four wheel drive) motion. Based on recent advancements[1] in autonomous vehicles using deep reinforcement learning methodologies[2], I offer a framework for driving the automobile while drifting along a track.

1 Objective

This project's goal is to create a deep reinforcement model for regulating the car's steering angle and throttle valve while driving around curves. The framework's implementation will be evaluated using a variety of reinforcement algorithms, with a focus on policy and off-policy approaches.

Algorithms like Proximal Policy Optimization (PPO) are an on-policy framework that works well with fresh data only after producing new samples for each policy update, which is a disadvantage when running in a new environment.

Off-poicy approaches, such as Deep Deterministic Policy Gradient (DDPG), perform well with fresh data since they rely on observations from the experience replay buffer.

The approaches stated above work well in the setting, although these are hyperparamater sensitive and require a long time to converge. The soft-actor critic comes to the rescue in the aforementioned situations.

2 Methodology

The following section details the methodologies and dependencies used in the project.

2.1 Control Method

The primary control policy is to control drift by following a trajectory, which we may refer to as a trajectory following task. The goal of this project is to steer the car along a path and drift through curves.

State Variables

Input State: Steering angle, throttle, velocity (x,y), slip angle (angle between route direction and velocity direction), and car orientation relative to the reference trajectory.

Output State : Steering angle and throttle valve

2.2 Proposed Algorithm

Our model will be trained using the Soft-Actor Critic (SAC)[3], which optimizes the error loss (anticipated return - prediction) and maximizes entropy using an off-policy learning strategy to perform better in continuous domains.

Control policy is the actor in SAC, while value and Q-network will function as critics.

The basic goal of the actor is to maximize reward while minimizing entropy (measure of randomness in the policy - more exploration).

The model is initially trained for several epochs, and the trained state transitions are saved using the best control policy. Following training, we use the experience replay buffer to select and perform an action at random.

2.3 Software and Simulation Requirements

Software Packages : Pytorch , Pygame, Python , Ubuntu

Simulation Environment: CARLA, a open-source simulator for autonomous driving using Unreal Engine will be used as simulator for training and validation.

Dataset : We need pre-defined maps which can be used from Road Runner, which will create a environment(track) used for testing self-driving simulation.

2.4 Phase-1 Report Status

This section details about the current status of this proposed algorithm and explains about the current simulation environment.

Simulation Environment

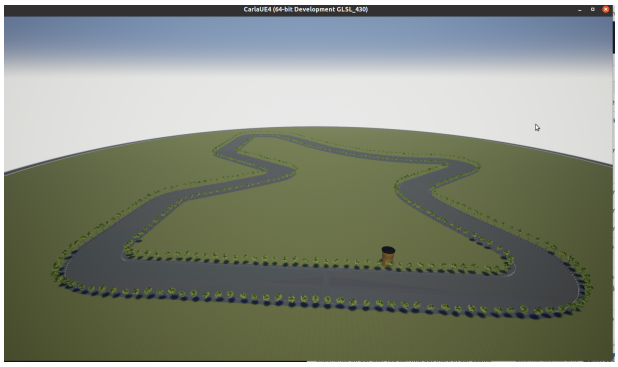

Figure 1. Sample Track which will be used for training

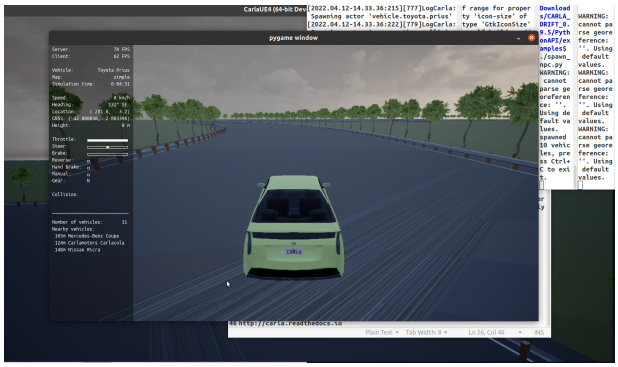

Figure 2. Sample Vehicle which will be used for training

I have been testing with different maps which are having sharp turns as to showcase the drifting of the vehicle. I have found about 5-6 tracks which will be suitable for this simulation.

Please refer to the github repository for setting up the simulation environment and simulating the above described map.

Please refer to figure - 1 for the simulation of the track in CARLA simulator. I will update all the steps to install and run this simulator with the current map in the GitHub repository.

This track will be used for training purpose and for testing we will be using a different map which will be similar to this one but with modifications. I will be using different vehicles to showcase the algorithm performance. Please refer to figure-2 to show different vehicles used for training.

Dataset Pre-processing

For this phase, I was planning on noting down all the trajectories values for each track(one vehicle right now) and keeping it as a reference to test out later when I run my model using the proposed algorithm. The parameters

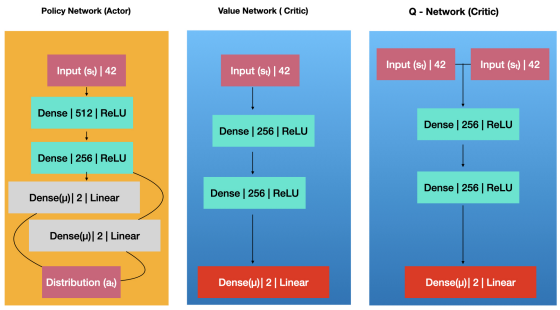

Figure 3. Network Structure for Policy and Value Networks.

which I will be considering are world coordinates(x,y) , heading angle, velocity(x,y) , slip angle and steering angle. These will be different for different vehicles so I'll be noting down only for one vehicle.

I have considered the steering and throttle values to be in range with the Carla simulator normalized structure and made slight modification to showcase high speed drifting without rolling over.

Algorithm Development

I have proposed Soft Actor Critic algorithm which will be used to control the throttle and steering angle during high speed driving around a track.

I have taking reference from the original paper of soft actor critic and a recent implementation of autonomous drifting using other off-policy networks. [4] for training the algorithm.

Please refer to the psuedo algorithm for soft actor critic mentioned below. I have selected the below Deep Neural Network structure for each of the network - Policy,Value and Q-network.

Algorithm 1 Soft Actor Critic Algorithm

Require: Replay Buffer B, total number of transisitons N , threshold η , Number of Updates λ
Result: Optimal Control Policy π^*_ϕ
Initialize the parameters of all the networks.
$D \leftarrow 0, N \leftarrow 0, s_t \leftarrow s_0$
for each episode **do**
  while $s_t \neq s_T$ **do**
    action π^*_ϕ , s_{t+1} $P(s_{t+1}|s+t, a_t)$;
    D $\leftarrow ReplayBuffer(s_t, a_t, r(s_t, a_t), s_{t+1})$;
    N $\leftarrow N + 1$,$s_t \leftarrow s_{t+1}$
  end while
  if $N >= \eta$ **then**
    Update all the networks λ times:
  end if
end for

Please refer to the Github Repository for the current

status of the project.
Github Repository

2.4.1 Phase - 2

1. Reference Trajectories have been computed and the values are being used for training the data.In the coming weeks, I'll be implementing with different vehicles and test it out with different tracks.
2. Adding the data to tensorboard for further referencing during the training process to understand various loss parameters and how they are performing under different constraints.
3. Add a smoothing condition to the action so as to avoid jerks and disrupt motions.
4. Simulating the trained algorithm and submit the report and presentation.

References

[1] Mark Cutler. Autonomous drifting using simulation-aided reinforcement learning. In *2016 (ICRA)*, pages 5442–5448, 2016. doi:10.1109/ICRA.2016.7487756.

[2] T. Bécs L. Orgován. "autonomous drifting using reinforcement learning". In *2021 Period. Polytech*, pages 292–300, 2021. doi:doi.org/10.3311/PPtr.18581.

[3] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018. URL `http://arxiv.org/abs/1812.05905`.

[4] Peide Cai, Xiaodong Mei, Lei Tai, Yuxiang Sun, and Ming Liu. High-speed autonomous drifting with deep reinforcement learning. *CoRR*, abs/2001.01377, 2020. URL `http://arxiv.org/abs/2001.01377`.