Self-driving vehicle in Duckietown environment
Duckpropagation

Attila Pethő **Farkas Olivér István** **Faragó Gyula**

Abstract

English abstract

Kivonat

Magyar kivonat, magyar betűk , üúó

1 Introduction

Self driving vehicles are clearly the future of transportation. If you look at waymo's [**?**] safety report you can see two things. First it is obvious that AI driven vehicles are more safe than a human driven ones. The other thing which makes development hard is the amount of data needed for the training of the AI. The report mentions that the AI used in the cars have learned from over 20 million driven miles. It is quite big number and an astounding achievement given the fact that the car fleet only had 8 bigger accident during this time. The report shows that all accident had a similar cause, human error. In three times the car was stationary and someone ran into it. I think it shows the potential of self driving vehicles.

For us 20 million miles driven seems a lot, but to train well an AI its not that much. This is the reason waymo chosen to train its AI in simulation as well. The driven mile is the simulation were 20 billion. That means in order to archive better results using a simulator is a good idea. Duckietown provides an excellent learning ground for those who want to understand the theory of reinforced learning, and apply the theory in real environment. It has started in a class at MIT in 2016. Duckietown has grown so much since then, it's now a worldwide initiative to learn AI and robotics. Our goal was to deep dive into the world of reinforced learning and develop efficient learning strategies for our duckiebots.

2 Methods

2.1 Reinforcement Learning

Reinforced learning differs from other machine learning techniques. It can't be put in supervised or unsupervised learning instead it creates its own separate category in machine learning. The reinforced learning model consist of two important element, the environment and the agent (Figure 1). The environment can be defined as everything that is not the agent and have no total control over it. The environment can be represented mathematically by a set of variables that define each state of the system. The set of all states called state spaces. The agent don't have full access to the state space, it can see only a part of it which is called observation. The agent can modify the current state through actions. The environment reacts to the action with potential state change and a reward.

The goal is simple, the agent have a predefined task in the environment and must complete it through actions. To archive its goal the agent have to do a three-step process: the agent interacts with the environment, the agent evaluates its behavior, and the agent improves its responses [**?**]. The evaluation is based on the previous rewards and the observations. The hardest part is the underlying algorithm in the agent which maps the observation and the rewards to a goal archiving action.

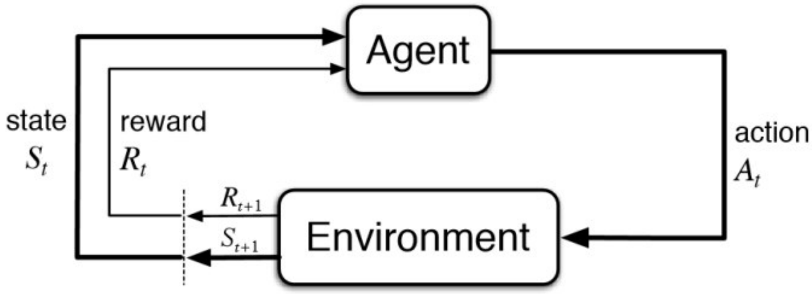

Figure 1: Illustration of Reinforcement Learning

2.2 A2C, PPO algorithms

We tried two promising algorithm for the task: A2C and PPO.The implementation of these algorithms were done by the stable baselines library [?] so we only had to focus on reward and observation shaping.

2.3 Environment

The environment gives us an 640x480 RGB image. We have to preprocess this image, because feeding the original images to the CNN would be a waste of resources because it makes training process much slower, and the network can learn from smaller images just as well.

2.3.1 Observations

Preprocessing actions are applied to the observations. These can be resizing the image, cropping the image, grayscale the image, segment the colors in it or stack the last n number of the frames.

Figure 2: (a)

Figure 3: (b)

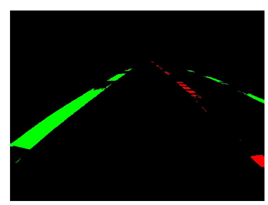

Figure 4: (c)

Figure 5: Observations from the environment. (a) shows a raw observation, (b) shows a grayscaled image, (c) shows a color segmentated image

2.3.2 Actions

Since the vehicle doesn't have turn-able wheels, the turning must be archived by modifying the speed of the individual wheels. In the real life this speed is expressed as the PWM (Pulse Width Modulated) value of the applied voltage on the motor. For simplicity this value is normalized to 1 both int the simulation and the real life. For easier calculation the PWM value is not calculated dynamically it is statically coded at fix values which equates to three individual state. These states are forward, right turn, left turn.

2.3.3 Rewards

Just like any other engineering problem choosing optimal rewards are tricky and require lot of testing and empirical estimation. A poorly chosen reward function can be exploited by the agent. In our first test runs the agent got reward for staying on the path. The reward was increasing and we got long episodes which looked promising. After examining the data we discovered that the agent was rotating around in one place. It realized that the easiest way to maximize reward was not what we initially intended as a goal. To correct this we had to take in account the angle ,position from the middle and speed of the agent.

2.4 Evaluation

There are many ways to measure the performance of the trained agents, and for the sake of simplicity we chose two metrics for this purpose: survival time (in timesteps) and rewards received from the environment. Survival time tells us how many steps the agent took before either it went out of the road or the episode finished. A well-trained agent should stay on the road for the whole evaluation episode. Altough this provides valuable information, this metric does not tell much performance on its own. Hence the rewards are also stored and evaluated to gain richer knowledge about the agents perfomance. The models were trained on the 'zigzag_dists' map, because it offers a good balance between left and right turns and straight sections. In order to test how well the models learned to generalize, evaluation was done on three different maps: 'zigzag_dists', 'loop_empty', and 'udem1'.

2.5 Hyperparameter Optimization

Reinforcement learning models are extremely sensitive to hyperparameter settings. A poor choice of hyperparameters can cause unstable performance or it could prevent convergence completely. Findig the right parameters for training can be a difficult job, and it is usually not a good idea to start tuning the parameters manually. Instead you can use an optimization library, that sets up sophisticated experiments to find the right hyperparameters for your usecase.

For our project we used the Optuna library, which offers several tools to dynamically explore the hyperparameter-space in a define-by-run style. The library features state-of-the-art alorithms for sampling and pruning, easy parallelization of experiments and visualization tools. In order to optimize our models, we used the Tree-structured Parzen Estimator (TPE) for sampling and the Median pruner for early termination of less promising trials. The studies were conducted with a budget of 100 trials and a maximum of 50000 steps, and evaluations at every 10000 steps. For optimizating the PPO model, we used an RTX 6000 workstation with 24GB VRAM and 46GB RAM, but for the A2C study, we had to use a GTX1060OC with 6GB of VRAM and 16GB RAM which was not enough to finish the study, so in that case the results are based on 47 finished trials. Table [TODO] shows the optimized hyperparameters for the two models.

Table 1: Optimized model hyperparameters

(a) PPO

Learning rate schedule	linear
Learning rate	0.8764
Batch size	256
n_steps	32
Entropy coef.	6.5699
Clip range	0.2
Epochs	5
Gae. lambda	0.92
Max grad. norm.	0.8
vf coef.	0.2425
Activation	relu

(b) A2C

Learning rate schedule	constant
Learning rate	0.1324
Norm. advantage	False
n_steps	64
Entropy coef.	0.0015
Use RMS prop.	False
Gae. lambda	0.98
Max grad. norm.	0.7
vf coef.	0.7373
Activation	tanh

3 Results

The PPO and the A2C agents were trained in the gym-duckietown simulator, with concatenating the last 3, grayscaled observations, and using the 'steering' actions and the orientation rewards described in subsubsection 2.3.2 and subsubsection 2.3.3. The agents were trained on the 'zigzag_dists' map for one million steps, using only 1 environment. Each episode lasted a maximum of 500 steps, because this proved to be enough to take at least a full round on the map for a reasonable agent. In order to examine the effects of the hyperparameter optimization, first we trained the models with mostly the default SB3 parameters, the only exception was the learning rate, which was set to a constant 0.0005 value. This was ment to be a baseline training to compare it with the optimized models. The main difference - besides the optimized parameters - was the use of 4 parallel environments and the 2 million steps long training. Table 2, Table 3 and Table 4 shows the evaluation results on the different maps.

Table 2: Evaluation results on the ’zigzag_dists’ map

	PPO (default)	A2C (default)	PPO (optimized)	A2C (optimized)
Mean episode reward	11.51	74.24	128.19	74.24
Mean episode length	71.5	308.0	452.9	308.0

Table 3: Evaluation results on the ’small_loop’ map

	PPO (default)	A2C (default)	PPO (optimized)	A2C (optimized)
Mean episode reward	14.83	61.21	93.78	61.21
Mean episode length	69.8	219.5	311.6	219.5

Table 4: Evaluation results on the ’udem1’ map

	PPO (default)	A2C (default)	PPO (optimized)	A2C (optimized)
Mean episode reward	6.01	98.27	70.79	98.27
Mean episode length	38.2	359.6	262.7	359.6

4 Conclusion