Duckietown

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this project we are training Deep Reinforcement Learning agents to drive small robots called Duckiebots in the Duckietown environment. There are four challenges in the environment:

•**LF** - simple lane following
•**LFV** - lane following with vehicles
•**LFI** - lane following with intersections
•**LFVI** - lane following with vehicles and intersections

In order to conquer these challenges, autonomous driving agents are first trained in a simulator (gym-duckietown) and then the trained agents performance are also tested in the real environment on real Duckiebots.

1 Introduction

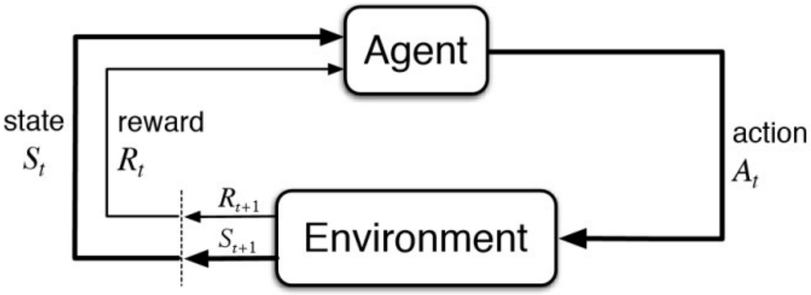

Figure 1: Illustration of Reinforcement Learning

Duckietown 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.

2 Installation

Requirements: git installed on your computer

You need to clone the following repository:

```
https://github.com/attila-petho/DuckPropagation
```

If you are done with that, change directory into DuckPropagation with the following command:

```
cd DuckPropagation/
```

After that you need to run the script that sets up the environment:

```
bash env_setup.sh
```

3 Methods

3.1 RL

3.2 A2C, PPO algorithms

3.3 Environment

The environment gives us an 640x480 RGB image which looks like the following: 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. We did the following preprocessing steps:

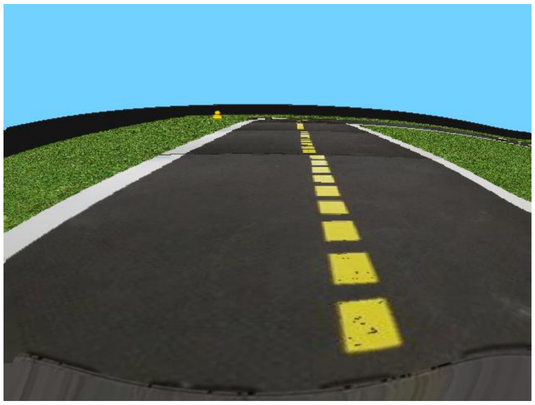

Figure 2: Observation of the environment without preprocessing

1. Resizing
2. Cropping
3. Color segmentation or Grayscaling
4. Normalization
5. Frame stacking

3.3.1 Observations

In this section we will present our environment wrappers which return the observations.

•ResizeFrame

With this wrapper we are downscaling the images from their original size (480x640x3) to (84x84x3). The smaller dimension makes the training of the neural network faster, and it still carries enough information for efficient training.

•CropFrame

In this wrapper we are cropping the useless information from the image, in our case it's the part above the horizon.

•GrayScaleFrame

Training time can be reduced by using grayscale images instead of RGB, while keeping the

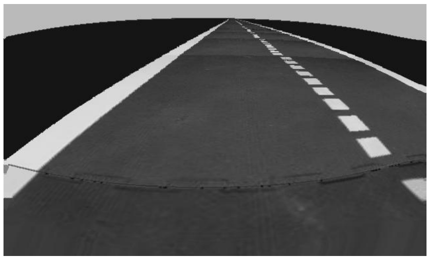

Figure 3: Observation from the environment with grayscaling

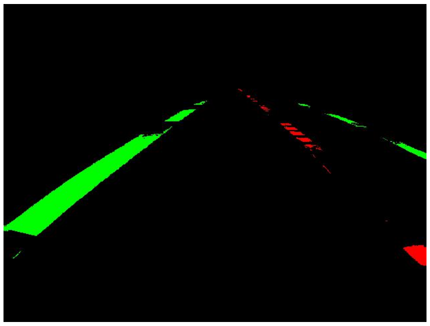

Figure 4: Observation from the environment with color segmentation

important information of the images. This wrapper should not be used in conjunction with the ColorSegmentFrame wrapper.

•ColorSegmentFrame
Here we are segmenting the different parts of the image so we can feed the neural network more useful information. The segmentation is done using intervals, we assigned the red channel for the white line, and the green channel for the yellow line. For lane following only these two information are useful for the neural network, so we assign black for everything else.
•NormalizeFrame
To make the training of the CNN easier and faster this wrapper normalizes the pixel values in the interval of [0,1]. Altough we implemented this wrapper, it is not used, because stable baselines does the input normalization automatically.
•StackFrame For better quality in training and more information we are concatenating the last n frames to form a time series, so the agent can percieve dynamic changes in its environment.

3.3.2 Actions

3.3.3 Rewards

3.4 Evaluation

3.5 Hyperparameter Optimization

4 Results

5 Conclusion

References

[1] András Kalapos, Csaba Gór, Róbert Moni and István Harmati. "Sim-to-real reinforcement learning applied to end-to-end vehicle control" arXiv:2012.07461 (2020) (`https://github.com/kaland313/Duckietown-RL`)