Deep learning in practice homework
Self-driving cars in the DuckieTown environment
DeepD(r)iveDucks team
Deep Learning a gyakorlatban házi feladat
Önvezető autózás a DuckieTown környezetben
DeepD(r)iveDucks csapat

Holló Áron
MSc in Autonomous Vehicle Control Engineering
Budapest University of Technology and Economics
Budapest
holloaron7@gmail.com

Tóth Tibor Áron
MSc in Autonomous Vehicle Control Engineering
Budapest University of Technology and Economics
Budapest
toth.tibor.aron@gmail.com

Hajnal Bálint
MSc in Mechatronics Engineering
Budapest University of Technology and Economics
Budapest
hajnal.balint.david@edu.bme.hu

Abstract

Our team used the reinforcement learning based DAgger (Data Aggregation) algorithm implementation for the DuckieTown autonomous driving environment in order to complete the lane following challenge. In this challenge an affordable bot has to follow the lanes in a simplified route environment. The DAgger algorithm is imitation learning based, which means that it is a framework for learning a behavior policy from demonstrations. Being a reinforcement learning method, it uses rewards to approximate the best policy, which is the expert policy in the DAgger algorithm. It uses a teacher agent, the aformentioned expert, that shows the learner agent the correct decisions - when needed - based on the current image input data. We improved on the baseline algorithm by implementing custom hyperparameter tuning to find the best parameters for good results. The evaluation is executed in a simulation provided by Duckietown.

Csapatunk a megerősített tanuláson alapuló Dagger (Adat Egyesítés) algoritmus implementációját használta a DuckieTown autonóm vezetési környezet sávkövetési kihívásának teljesítésére. A feladat egy robot mozgásának vezérlése úgy, hogy kövesse a sávokat egy utakkal ellátott egyszerűsített környezetben. A Dagger algoritmus az imitációs tanításon alapul, ami azt jelenti, hogy korábbi mintákra épülő viselkedési stratégia megtanulására szolgáló keretrendszer. Mivel ez egy

Deep learning in Python and LUA language in practice (BME 2021), Budapest, Magyarország.

megerősített tanulási módszer, ezért jutalmakkal erősítjük meg a helyes viselkedést, ami a “szakértő” viselkedése. Ez a szakértő a tanító ágens, ami szükség esetén megmutatja a helyes döntéseket a bemeneti képadatok alapján. Az alap algoritmust továbbfejlesztettük egyedi hiperparaméter optimalizációval, hogy megtaláljuk azokat a tanítási paramétereket, amikkel a legjobb eredményt érhetjük el tanítás során. A kiértékelés a DuckieTown platform szimulációján keresztül történik.

1 Introduction

The DuckieTown platform is both a physical and virtual environment, where people can study machine learning and artificial intelligence for self-driving cars. The platform has a competition, where students from all over the world can compete on who can make the most successful DuckieBot. Competitors often train their machine learning models in a virtual environment, which has an equal looking physical counterpart where the models can be seen working in real life. The virtual environment provides a cheap and easily accessible way to conduct machine learning research. Due to every solution being based on computer vision, the results from the virtual environments carry over to the real world pretty consistently.

2 Topic description

There are several implementation methods for the DuckieTown challenges due to the fact that computer vision is a very wide topic, and also there are several challenge types. We wanted to create an implementation for the lane keeping challenge, where the goal is to keep the so-called DuckieBot on the track in the randomly generated environment and achieve the furthest distance in a given time.

We chose an imitation learning algorithm called "DAgger", that is "Data Aggregation" for our implementation. Imitation learning is a reinforcement learning method, which uses an expert solution to reward the agent. In a self-driving car environment it is really hard to manually design a reward function, and the imitation learning solves this problem. The expert provides a set of demonstration to the agent to learn from. The agent tries to learn, mimic the optimal policy by imitating the expert's solution. The problem with a simple imitation learning algorithm is that the training data is not identically and evenly distributed, which means that if the agent is not following the expert's trajectory properly, and deviates from it, then it can reach a state from where it can not recover to the optimal track, because the expert never experienced that state. Robustness is not guaranteed in this solution. This is what the "DAgger" or Data Aggregation algorithm offers a solution for. The agent interacts with the environment, but sometimes the expert takes over control to help recovery. They are changed randomly hoping that the expert will help in recovering, therefore learning takes place.

Due to the Data Aggregation algorithm baseline being available to us, our main goal was to study the system and improve it. We executed custom hyperparameter optimisation on an existing implementation of the algorithm.

2.1 Previous solutions

The challenges are solved with a wide range of approaches. Our focus was mainly on "DAgger" algorithm solutions.

The "DAgger" algorithm was initially developed by Stephane Ross and his co-authors in 2011 [10].The baseline "DAgger" solution provided for the competitors on the official website - which we tried to improve - is based on the paper by Diaz and Manfred [4], the previously mentioned original algorithm [10] and a solution made for drones to drive on the streets [7]. They developed an example of a system which aims to help visually impaired people cross intersections by capturing the knowledge of sighted people. They also developed a theoretical and experimental framework for consistent performance in different environments. More research around DAgger algorithm can be read in these papers: [9] [2] [3] [6] [1] [5] [8].

3 System plan

Due to using an existing solution, we first needed to study its system in order to improve it. The program is written in Python, and it uses the PyTorch machine learning framework.

`train.py` is the script, that starts the training based on the arguments passed in. We implemented our hyperparameter optimisation in `train_tune.py`, which is based on this script.

The `dronet.py` file contains the architecture of the neural network based on Dronet [7]. Its a convolutional network consisting of a 5x5 convolutional filter, a 3x3 max pooling layer, 3x3 convolutional filters, a dropout filter and ReLU activation. The outputs are the steering angle and acceleration.

`pure_pursuit_policy.py` contains the teacher algorithm, which projects a point for the agent to follow.

`test.py` is used to test a model, after passing it in to the script.

`dataset.py` optimizes RAM usage by storing arrays on disk.

`neural_network_policy.py` is a wrapper to train a neural network model. It takes a pytorch mode, and an optimiser as inputs among others, and uses the expert actions to predict.

`dagger.py` is the "DAgger" algorithm, which is used to mix policies between learner and expert.

`iil_learning.py` contains the main imitation learning algorithm.

4 Implementation

4.1 Data acquisiton, preparation

The teaching is done by an expert agents actions this is how the data is acquired for learning. The agent can either be controlled by a human in the virtual simulator, or by a preprogrammed algorithm. The data received by the learner are the images in 120x160 size from the DuckieBots' vision, and the expert actions.

4.2 Teaching

The learner agent tries to follow the teacher, and predict the correct actions. If the learner starts to deviate from the track too much, then the control is given back to the expert. It is controlled by thresholds. The convergence angle threshold is for the angle deviation between the expert and the learner. The convergence distance threshold is for the distance between the learner and expert's trajectory.

4.3 Evaluation

We calculated the average loss after each episode to compare them. The loss function takes into consideration the velocity and the steering angle of the learning agent and the expert. For the velocity, the binary cross entropy loss is used with a sigmoid function. A decay factor is also used to for the velocity loss. For the steering angle the mean squared error is calculated. The cumulated error is their added loss.

We implemented a grid search hyperparameter optimization for the teaching. The parameters space we considered to iterate through was:

- episode number: 10, 15
- horizon: 64, 128, 256
- learning rate: 1e-1, 1e-2, 1e-3, 1e-4
- learning rate decay: 0.5, 0.6, 0.7, 0.8
- batch size: 16, 32.

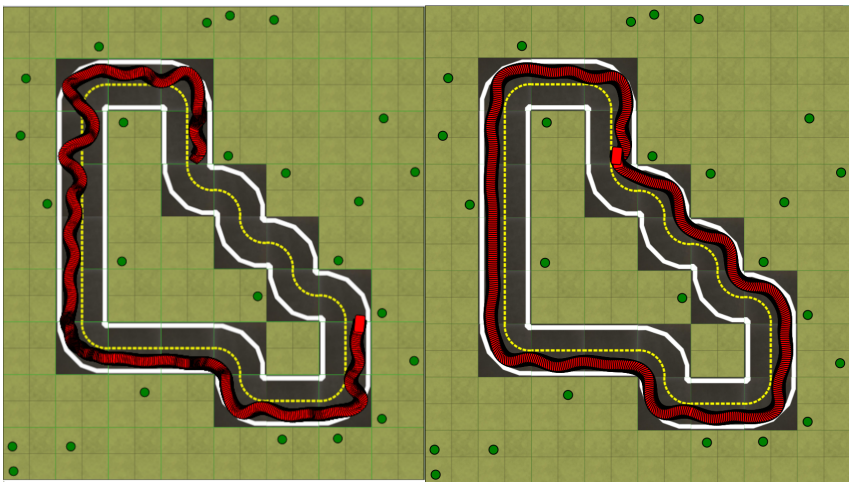

Figure 1: The best result from the baseline solution (left) and the hyperparameter optimisation (right)

Table 1: Comparing Duckietown metrics for each solution

Metrics	Baseline	Hyperparameter optimised
Driving distance [m]	10,1	12,5
Survival time [s]	60	60
Heading deviation [°]	15.14	10.46

The best results of the DAgger baseline solution and our hyperparameter optimized solution for this map can be seen on figure 1. Both of the solution were able to stay on the track during the run, but as it can be seen in table 1, the running distance - which is the main deciding factor - differs. We were able to run the duckie bot around 1/5th further on the track. The heading deviation also outperformed the baseline, which means the robot was able to run much more smoothly, with less waddling.

4.4 Testing

Due to the fact that we are not working with existing data, but rather "live" data created on the fly, our solution didn't use testing.

5 Conclusion, future plans

We spent a lot of time figuring out how the baseline works, where can we make modifications to implement what we wanted. Setting up the docker image and a cloud based computer with a jupyterlab for the training, testing, evaluating needed much more time for us than anticipated, because we are not really experienced with these systems. We set up Tensorboard for our hyperparameter and metric logging, so we could see the loss functions for each episode and the hyperparameter set of each models. Complications also occured from large files that the tensorboard produced during plotting, which filled up the memory of the cloud computer. These things set us back on time unfortunately, therefore we didn't really had time to implement all our plans.

Despite being successful at the lane following task, there are always more ways for improvement. It would be beneficial to implement hyperparameter optimisation over a wider range of possible parameters to get more accurate results. Another improvement could be better teacher algorithm, or a different neural network architecture.

References

[1] Elers Andreas. Continual imitation learning: Enhancing safe data set aggregation with elastic weight consolidation, 2019.

[2] Yunus Bicer, Ali Alizadeh, Nazim Kemal Ure, Ahmetcan Erdogan, and Orkun Kizilirmak. Sample efficient interactive end-to-end deep learning for self-driving cars with selective multi-class safe dataset aggregation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2629–2634. IEEE, 2019.

[3] Yuchen Cui, David Isele, Scott Niekum, and Kikuo Fujimura. Uncertainty-aware data aggregation for deep imitation learning, 05 2019.

[4] Manfred Ramon Diaz Cabrera. *Interactive and Uncertainty-aware Imitation Learning: Theory and Applications*. PhD thesis, Concordia University, 2018.

[5] Ahmed Hussein, Mohamed Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys*, 50, 04 2017.

[6] Farzad Kamrani, Andreas Elers, Mika Cohen, and Amir H. Payberah. Mariodagger: A time and space efficient autonomous driver. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1491–1498, 2020.

[7] Antonio Loquercio, Ana I Maqueda, Carlos R Del-Blanco, and Davide Scaramuzza. Dronet: Learning to fly by driving. *IEEE Robotics and Automation Letters*, 3(2):1088–1095, 2018.

[8] Montaser Mohammedalamen, Waleed D. Khamies, and Benjamin Rosman. Transfer learning for prosthetics using imitation learning. *CoRR*, abs/1901.04772, 2019.

[9] Aditya Prakash, Aseem Behl, Eshed Ohn-Bar, Kashyap Chitta, and Andreas Geiger. Exploring data aggregation in policy learning for vision-based urban autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11763–11773, 2020.

[10] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.