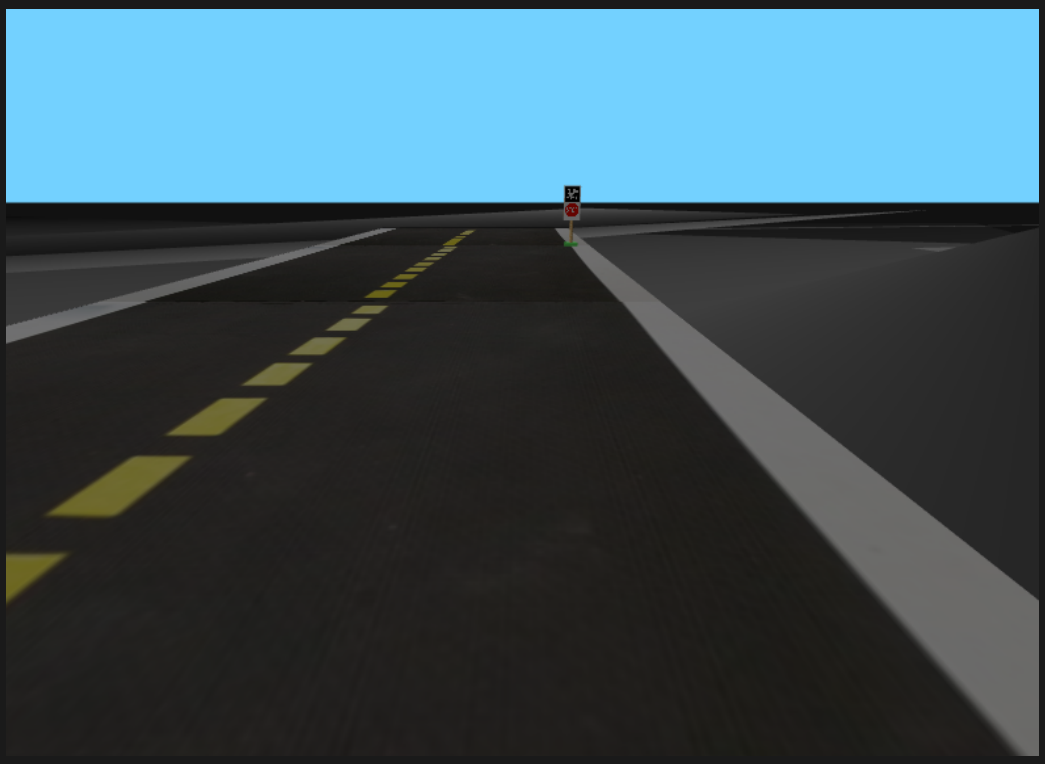
Teaching a Virtual Duckietown Agent to Stop



Sebastiaan C. Aflaki

Layout: typeset by the author using LaTeX.
Cover illustration: Gym-Duckietown Simulator (Chevalier-Boisvert, Golemo, Cao, Mehta, & Paull, 2018)

Teaching a Virtual Duckietown Agent to Stop

Sebastiaan C. Aflaki
11230851

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
Dr. A. Visser

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904
1090 GH Amsterdam

April 15, 2021

Abstract

A fully self-driving car (SDC) is a vehicle that can keep track of its surroundings and navigate without human assistance. Although SDC's are already being utilized, safety concerns limit public use. One of those concerns lies with the handling of traffic signs. As human drivers ignoring stop-signs often leads to dangerous situations, it is of the utmost importance that SDC's handle them correctly. This study attempts to teach a reinforcement learning agent proper stop-sign behaviour in order to increase the safety of self-driving cars.

Keywords: Reinforcement Learning, Proximal Policy Optimization, Stop-Sign Behaviour, Autonomous driving.

Contents

Chapter 1

Introduction

1.1 Context and topic

A fully self-driving car (SDC or autonomous car) is a vehicle that can keep track of its surroundings and navigate without human assistance. For over fifty years, self-driving vehicles have been a focal point of study in artificial intelligence and it is fair to assume SDC's are no longer merely a concept. Although completely autonomous vehicles are not yet widely available to the public, several applications of fully driverless vehicles are presently on the road for public usage. Today's autonomous cars navigate their environment by utilizing a variety of tools including Lidar, Radar, GPS and, most critically, computer vision to intpret this data. Google's Waymo cars, for example, which currently serve as a driverless taxi service, utilize Lidar, radar, computer vision and numerous other sensors to map, scan, anticipate and respond to traffic environments.

SDC development has always been primarily motivated by the need to increase road safety. Studies show that human error was found to be responsible for 94% of

all traffic collisions in the United States (Singh, 2015). As fully autonomous cars take away all vehicle control from human drivers, they are believed to effectuate a substantial decrease in traffic accidents involving human drivers(Policy, 2016). However, before SDCs are made available to the public, the difficult task of creating control systems robust to all kinds of traffic situations and unforeseen events must be perfected. Scenarios such as intersections, can easily create dangerous situations if navigated wrongly. Due to the wide variety of intersection configurations, this can be an exceptionally difficult subject to generalize. Stop-signs, which urge vehicles to come to a complete stop regardless of whether or not there is oncoming traffic, are an important part of traffic regulations. In the United States the primary method of regulating traffic at intersections is through the use of stop-signs(Retting, Weinstein, & Solomon, 2003). A study by Retting et al. collected data on two-way stop-sign-controlled intersection crashes in four U.S. cities and found that out of 1.788 crashes about 70% consisted of stop-sign violations. Furthermore, about 48% of all crashes at stop-sign-controlled intersections were related to drivers ignoring stop-signs. (Retting et al., 2003) To ensure that autonomous vehicles are beneficial to road safety, they must be capable of correct stop-sign behaviour.

1.2 Scope and focus

There are several approaches to teaching a self-driving agent to obey stop-signs using machine learning. One of the more recent and interesting approaches in self-driving car research is Reinforcement Learning (RL). Reinforcement learning is a subfield of machine learning which investigates how an agent might acquire the ability to accomplish goals in a complicated, unpredictable environment. (Kiran et al., 2021) In self driving car research, reinforcement learning has been used to tackle many important tasks such as traffic light control, lane following and obstacle avoidance. (Li, Xu, & Zhang, 2021) (Kalapos, Gór, Moni, & Harmati, 2021) (Saavedra-Ruiz, Morin, & Paull, 2022) However, there has been little to no research into using reinforcement learning to teach an agent stop-sign behaviour specifically. To achieve this, this thesis utilizes the clipped variant of the proximal policy optimization reinforcement learning algorithm. (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017) (*Proximal Policy Optimization — Spinning Up documentation*, 2018)

1.3 Research question

The purpose of this study is to understand how reinforcement learning can benefit the process of developing safe driver-less cars. This will be accomplished by addressing the challenge of teaching a reinforcement learning agent stop-sign behaviour. Therefore the research question is: *Can an agent accurately learn stop-sign behavior using reinforcement learning?* With considerable hyper-parameter adjustment, it is reasonable to expect that a reinforcement learning agent may be taught to exhibit proper stop-sign behaviour. As is apparent in literature surrounding policy gradient methods. (Paul, Kurin, & Whiteson, 2019) (Khadka & Tumer, 2018)

Chapter 2

Theoretical background

2.1 Duckietown

The Duckietown foundation offers a fun and accessible solution for AI and robotics education and research. (The Duckietown Foundation, 2016) Their small robotic cars (DuckieBots) are equipped with a camera and a GPU, which offers a relatively affordable alternative for autonomous vehicle research using real-world cars.

Figure 2.1: The DuckieBot version DB19

Additionally, they offer the resources to build small scale roads with obstacles, traffic equipment and various sensors creating endless possibilities for simulating complex problem environments. Along with their physical DuckieBots, they provide a convenient simulator for further testing and training purposes.

2.1.1 AI Driving Olympics

Furthermore, since 2018 Duckietown has hosted the AI Driving Olympics (AI DO) twice a year at ICRA (International Conference on Robotics and Automation) (IEEE Robotics and Automation Society, 1984) and NeurIPS (Neural Information Processing Systems Conference) (Neural Information Processing Systems Foundation, 1987). The AI DO challenges range from simple single robot lane following (LF) tasks to multi-robot lane following with vehicles, pedestrians and intersections (LFIVP).

2.1.2 Simulator

Gym-Duckietown is a driving simulator based on OpenAI's gym platform (OpenAI, 2016). While Gym-Duckietown is a low-fidelity simulator, solutions have been shown to work well on real-world Duckietowns due to their comparable simplicity and in-simulation techniques such as domain randomization, accurate camera distortion, and differential-drive physics. (Chevalier-Boisvert et al., 2018) The simulator is highly customizable with options for various obstacles, road map layouts and traffic signs.

Figure 2.2: The Duckietown Gym Simulator

2.2 Reinforcement learning

As briefly mentioned in chapter 1, reinforcement learning is a subfield of machine learning which investigates how an agent might acquire the ability to accomplish goals in a complicated, unpredictable environment. Contrary to the machine learning branch of supervised learning, an RL agent learns by the consequences of its actions rather than through explicit instruction, and it chooses its actions based

on prior experiences (exploitation) as well as on new alternatives (exploration), which is similar to trial and error learning. The reinforcement signal received by the reinforcement learning agent is a numerical reward that encodes the success of an action's result, and the agent learns to choose behaviours that maximise the cumulative reward over time. In a variety of tough contexts, reinforcement learning algorithms have begun to show promise. While reinforcement learning has a long history, before recent improvements in deep learning, it necessitated extensive problem-specific engineering. DeepMind's Atari achievements (Mnih et al., 2013), BRETT from Pieter Abbeel's group (Yang, 2016), and AlphaGo (Holcomb, Porter, Ault, Mao, & Wang, 2018) all employed deep reinforcement learning algorithms that made few assumptions about their environment and are thus applicable in various contexts. However, two issues impede the progress of RL research:

First off, the requirement for more accurate benchmarking. In supervised learning, development has been accelerated by the availability of huge labelled datasets such as ImageNet (Stanford Vision Lab, Stanford University, Princeton University, 2020). The closest analogy in RL would be a broad and varied selecion of environments. However, existing open-source collections of RL environments lack diversity and are sometimes difficult to set up and use. The second issue that hinders advancement in RL is the limited standardization of the environments utilised in publications. The complexity of a task can be substantially altered by small changes in the issue formulation, such as the reward function or the definition of the action space. Therefore, reproducing published research and comparing results from various research papers is made significantly more complex (OpenAI, 2016). To address these concerns, OpenAI, an open source AI research and deployment business, developed the "Gym" toolkit, a very useful development and comparison toolkit for reinforcement learn- ing algorithms.

2.2.1 Policy Gradient Methods

Policy gradient methods are at the heart of current state-of-the-art reinforcement learning models. Policy Gradient methods function by calculating an estimator and inserting it into a stochastic gradient ascent algorithm. (Schulman et al., 2017) Generally speaking, the most widely used gradient estimator is of the form

$$\hat{g} = \hat{\mathbb{E}}[\nabla_\theta log \pi_\theta(a_t \mid s_t)\hat{A}_t]$$

where π_θ is a stochastic policy and $\hat{A}_t$ is an estimator of the advantage function at time-step t. The expectation $\hat{\mathbb{E}}[...]$ denotes the empirical average across a finite batch of sample data in a sampling-optimization algorithm. In turn, the estimator $\hat{g}$ is produced by differentiating the objective (Schulman et al., 2017)

$$L^{PG}(\theta) = \hat{\mathbb{E}}[log\pi_\theta(a_t \mid s_t)\hat{A}_t]$$

.

A smart optimization function is obtained by multiplying the log probabilities of the policy's output and advantage function. If the advantage is positive, implying that the agent's last action during the sample trajectory resulted in a higher-than-average return, the policy gradient will likewise be positive to improve the chance of repeating said actions when confronted with a similar situation. In turn, if the advantage is negative, the policy gradient will be negative in order to decrease the chance of repeating similar actions.

While it may seem beneficial to continuously optimize loss function $L^{PG}(\theta)$, parameters will regularly update far outside of their range, as a result, policy updates become too large.

Chapter 3

Method

This chapter describes the experiments done and substantiates the research's conclusions. The experiment of teaching a virtual agent stop-sign behaviour was conducted using a Proximal Policy Optimization (PPO) (Schulman et al., 2017) reinforcement learning algorithm in a Duckietown simulator environment. A selection of the various trained PPO models were evaluated through testing on new situations in addition to comparing training data.

3.1 Algorithm

This study utilizes a Proximal Policy Optimization algorithm from OpenAI's Stable Baselines 3(Hill et al., 2018). Due to a similar underlying framework, Open AI's PPO method can easily be implemented into the DuckieTown-gym simulation environment.

To avoid excessively large and damaging policy updates that occur in vanilla policy gradient methods as described in section 2.2.2, Trust Region Policy Optimization was developed. (Schulman, Levine, Abbeel, Jordan, & Moritz, 2015) The authors of this study created a method to restrict the policy gradient step so that it does not deviate too far from its original policy.

$$\hat{g} = \hat{\mathbb{E}}_t\Big[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\hat{A}_t\Big] \longrightarrow \begin{array}{ll} \underset{\theta}{\text{maximize}} & \hat{\mathbb{E}}_t\left[\dfrac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] \\ \text{subject to} & \hat{\mathbb{E}}_t[\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta. \end{array}$$

Figure 3.1: (Schulman et al., 2017)

Apart from the log operator being replaced with a probability term based on the action of the current and the previous policy, TRPO is similar to vanilla policy gradient methods.

Furthermore, the TRPO method includes a restriction on the KL distance (or relative entropy), preventing the gradient step from deviating too excessively from the previous policy. This ensures that the gradient remains in a range where we can be confident that previous functionality is kept, hence the etymology of the term "trust-region." On the other hand, KL restrictions are known to add cost to the optimization process, occasionally resulting in poor training behaviour.

As a way to streamline the optimization process while maintaining the advantages of TRPO, Proximal Policy Optimization (PPO) (Schulman et al., 2017) has been developed. PPO presents two main variants to achieve similar if not better results than TRPO methods: the PPO-Penalty method and the PPO-Clip method. Much like TRPO methods the PPO-Penalty method works with KL-constrained policy updates but instead penalizes objective function KL-divergence rather than imposing a hard restriction while automatically scaling the penalty parameter throughout training. The PPO-clip method however, does not include a KL-divergence component or restrictions at all. To deter a new policy from straying too far from the previous policy the PPO-clip method utilizes a clipped objective function. (*Proximal Policy Optimization — Spinning Up documentation*, 2018)

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$$

In this function, the expectation is computed over at least two terms: the conventional policy gradient objective and the clipped policy gradient objective. The critical component comes from the second term, which truncates a standard PG objective via a clipping operation between 1-epsilon and 1+epsilon, where epsilon is the hyper-parameter.

The graph on the left depicts the positive advantage: the circumstance in which a selected action had a positive effect on the outcome. The loss function flattens out on the graph when the r becomes too large or when an action is significantly more likely under the current policy than it was under the previous policy. The same holds true for the graph on the right under the circumstance the estimated advantage is negative. When r approaches zero, the loss function flattens out,

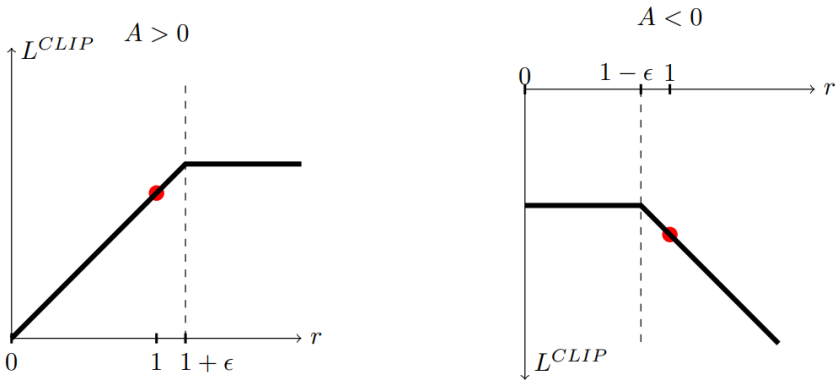


Figure 3.2: Plots of a single time-step of the surrogate function L^{CLIP} as of function of probability ratio r, for positive and negative advantages. (Schulman et al., 2017)

indicating that a selected action is significantly less likely under the present policy.

Additionally, the clipping method may also be used to correct policy flaws. For instance, the red dot in the right graph illustrates where the last gradient update increased the probability of the selected action significantly while simultaneously deteriorating the policy, as indicated by a negative advantage. To correct this, the clipping procedure will revert the gradient exactly the amount that was incorrectly added. In this instance, the first term in the min() operator is less than the second term, serving as a fallback. Contrary to PPO-penalty and vanilla policy gradient methods, PPO-clip does not require calculating additional KL constraints for gradient correction.

3.2 Training

The PPO agent is placed on a shorter customized version of the straight road map provided by DuckieTown including a stop-sign. The straight road map consists of seven 'road' tiles with a stop-sign at the start of the fourth tile.

The reinforcement learning model trains by getting the simulator image every time-step and predicting an action for that particular image. The agent has control over its left and right wheel drive, meaning every time-step, its action space consists of a continuous two-item array with values between 0 and 1: 0 meaning no drive and 1 meaning building up to max speed for that particular time-step. To simplify the problem one of two wheel drive values is used for both wheels reducing the action space to a throttle and excluding steering. For additional bounding pur-

poses, values beneath 0 were excluded resulting in the car only being able to drive forwards. Actions start of as random sampling of the action space and gradually become less random each policy update.

The goal is for the agent to drive up to the stop-sign and stop as close as possible to a set of coordinates encoding the ideal halting spot next to the stop sign. Every training episode, the agent controls the throttle for 500 time-steps which is equivalent to the time it would take an agent to drive to the end of the road at moderate speed. Each time-step the agent gets closer to the goal it earns a reward of 1, and vice versa, driving away from the goal earns the agent a reward of -1. To make sure the rewards are given accordingly every time-step the distance to the goal halting spot is compared to the distance of the previous step. As the PPO agent aims to maximize its reward per episode it must learn to avoid driving past the halting spot and thus, learn to stop in front of stop-signs.

3.2.1 Parameters

Below important parameters and their starting values will be shown. Hyper-parameters with an asterisk (*) have varied throughout the training process. Furthermore, each parameter will be shortly explained and additional environmental hyper-parameters will be exemplified.

Parameter	Value
Policy	CnnPolicy
Environment	gym-duckietown
Learning rate	0.0003
Number of Steps*	2.048
Batch size	64
Epochs	10
Clip range	0.2

Table 3.1: The hyper-parameters used for training PPO

Policy
The neural net used to train the policy.

Environment
Environment to learn from.

Learning rate
Determines how much is learned from positive experience each policy update.

Number of steps* (n_steps or horizon)
Determines the number of time-steps of collecting experience before each new policy update, and thus lowering this will result in more frequent policy updates.

Epochs
Determines amount of epochs used in stochastic gradient descent update.

Batch size
Size of batch used in stochastic gradient descent updates on all gathered trajectories for the specified number of epochs.

Clip range
Clip range is the ϵ parameter specified in section 4.1.

Of the models included in the final results, solely the models STEP500 and STEP1000 were trained using different n_step values, 500 and 1000 steps respectively.

Additional parameters

Early evaluation experiments found that the first few well-performing models learned to stop in the specified spot by "looking" at the stop-line instead of the stop-sign. From these first few models PPO4 was included in the results. To make sure subsequent models would not learn this behaviour, the stop line, which was previously included in a intersection tile behind the stop-sign, was excluded.

Secondly, both early stop-sign and stop-line models would often either linger close to the ideal stop spot accelerating at extremely low speed instead of coming to a complete stop lengthening the episode or continue driving after passing the halting place accidentally. To deter future models from behaving similarly, an additional rule was introduced to the environment that resets the environment anytime the agent's speed falls below a specified threshold. The disadvantage of this restriction was that agents began halting prematurely more frequently. With each subsequent model trained, the threshold was altered to find an optimal balance between premature stopping and the lingering behaviour. This speed threshold was first established with the model PPO10 and started at 0.03 m/s and was decreased to 0.01 m/s which improved performance slightly. Afterwards, the speed threshold was raised to 0.2 m/s which resulted in a clear deterioration

of performance. Simultaneously, the stop-sign in the simulator was enlarged to attempt to increase the "recognizability" and further improve performance. The stop-sign size was initialized at 0.08 and was later enlarged to a size of 0.12. Eventually, it was found that a speed threshold of 0.0055 m/s reduced the amount of premature stops while minimizing lingering behaviour and stop-sign violations.

3.3 Evaluation

To evaluate performance, trained models were tested by letting them predict the best actions for 20.000 time-steps for each of the three different stop-sign scenarios. As a way of validating training, the first scenario is identical to the training setting described in section 4.3. The second and third scenario are similar to the first: the "straight_road_train" map with seven 'road' tiles however, the stop-sign is placed near the end of the road at the start of the sixth road tile in the second "straight_road_far _test" scenario and at the start of the second tile in the third "straight_road_close_test" scenario. Performance was measured per episode and was divided into three categories. Premature stop: in the circumstance that the agent would stop before the stop-sign but outside a 0.3 meter distance margin from the ideal halting coordinates. Correct stop: in the circumstance that the agent would stop before the stop-sign and inside a 0.3 meter distance margin. Stop-sign violation: in the circumstance that the agent would drive past the ideal halting coordinates. In addition to experiment testing evaluation, a Tensorboard (Abadi et al., 2015) log setup was utilized during training allowing for additional training data of each model trained which can be found in appendix B.

Chapter 4

Results

In this chapter the test results from a selection of notable models (over a total of 50) will be shown through comparison tables. The following models were included due to their interesting differences in terms of results and parameters: PPO4*[1], PPO25, PPO30, PPO31, STEP500 and STEP1000.

4.1 Experiment Data

4.1.1 Train Evaluation

Model	Premature stop (%)	Correct stop (%)	Stop-sign violation (%)
PPO4*	16.66	**83.33**	0
PPO25	0	0	100
PPO30	28.43	57.84	13.73
PPO31	61.63	38.37	0
STEP500	63.16	9.47	27.37
STEP1000	23.33	70	6.67

Table 4.1: Evaluation performance of various PPO models on training road (stop-sign central)

Table 4.1 describes the results of the evaluation of various models on the "straight_road_train" map the models were trained on as described in section 3.2 and 3.3.

[1]* The model PPO4 was trained on a simulator map that included a stop line in addition to the stop-sign. After evaluation, it was discovered that the model had been trained on the stop line rather than the stop-sign. While learning to stop in front of the stop line was not the intended outcome, PPO4's performance was an intriguing result to demonstrate.

4.1.2 Test Evaluation Far

Model	Premature stop (%)	Correct stop (%)	Stop-sign violation (%)
PPO4*	50	**50**	0
PPO25	0	0	100
PPO30	47.3	43.24	9.46
PPO31	100	0	0
STEP500	79.1	10.45	10.45
STEP1000	51.28	44.87	3.85

Table 4.2: Evaluation performance of various PPO models on testing map (stop-sign far away)

Table 4.2 describes the results of the evaluation of various models on the "straight_road_far_test" map as described in section 3.3.

4.1.3 Test Evaluation Close

Model	Premature stop (%)	Correct stop (%)	Stop-sign violation (%)
PPO4*	0	**100**	0
PPO25	0	0	100
PPO30	0	72.34	27.66
PPO31	0	**100**	0
STEP500	0	23.08	76.92
STEP1000	0	85.44	16.02

Table 4.3: Evaluation performance of various PPO models on testing map (stop-sign close by)

Table 4.3 describes the results of the evaluation of various models on the "straight_road_close_test" map.

4.1.4 Analysis

Results show that from the models that actually learned to stop for stop-signs, the model STEP1000 performs best. STEP1000, trained with 1.000 n_steps, scores **70**% on the train map, **44.87**% on the far test map and **85.44**% on the close test map. PPO30, trained at the default 2.048 n_steps, comes in at second best scoring

57.84% on the training map, **43.24**% on the far test map and **72.34**% on the close test map. And lastly, the STEP500 model, which trained with 500 n_steps and performed extremely poorly, scoring **9.47**% on the training map, **10.45**% on the far test map and **23.08**% on the close test map. Another noteworthy outcome is the model PPO25, which was the only model trained for 75.000 time-steps. As can be seen from the figures A and B from appendix B.2, PPO25 performed extremely well until about 45.000 time-steps in, where it seemingly unlearned its policy and started driving at full speed at every state resulting in 100% stop-sign violations in all scenarios.

As previously established, PPO4 learned to stop for stop-lines instead of stop-signs and had the best performance overall, scoring **83.33**% on the train map, **50**% on the far test map and **100**% on the close test map.

Chapter 5

Conclusion

This thesis aims to understand the role of reinforcement learning in the process of developing safe driver-less cars and thus aimed to answer the question: "Can an agent accurately learn stop- sign behavior using reinforcement learning?"

The results indicate that the best-performing stop-sign-oriented models (PPO30 and STEP1000) do not perform well enough to qualify as having learnt accurate stop-sign behaviour. However, although model PPO4 is not a stop-sign oriented model, it shows great promise in its correct stopping scores in both the train and close maps. Moreover, as expected and demonstrated by tuning the n_steps parameter, the stop-sign size and the speed threshold: parameter tuning can have a considerable effect on model performance. Finally, all models appear to have difficulty anticipating stop-sign behaviour when the stop-sign is a long distance away. Thus, teaching an agent accurate stop-sign behaviour requires more parameter tuning in addition to a solution for long distance anticipation tasks.

Chapter 6

Discussion

The regularity with which stop-signs violations and premature stops occur would be deemed hazardous in comparable real life situations. Even though they are not yet suitable for use in driver-less automobiles, they give an intriguing foundation upon which to expand.

Although tough to pinpoint, the increased performance stop-line model PPO4 holds over its stop-sign equivalents may be explained by the difference in derivable information in the simulator image. In case of PPO4, the stop-line is a larger and more centered object than the stop-sign is, making it more visible even from a further distance. However, this gap may be bridge in future work by further tuning the n-steps, additional model parameters and the stop-sign size. Once a level of safety is established, subsequent study may involve domain randomization in order to translate this stop-sign research to actual DuckieBots.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from https://www.tensorflow.org/ (Software available from tensorflow.org)

Chevalier-Boisvert, M., Golemo, F., Cao, Y., Mehta, B., & Paull, L. (2018). *Duckietown environments for openai gym.* https://github.com/duckietown/gym-duckietown. GitHub.

Hill, A., Raffin, A., Ernestus, M., Gleave, A., Kanervisto, A., Traore, R., ... Wu, Y. (2018). *Stable baselines.* https://github.com/hill-a/stable-baselines. GitHub.

Holcomb, S. D., Porter, W. K., Ault, S. V., Mao, G., & Wang, J. (2018). Overview on deepmind and its alphago zero ai. In *Proceedings of the 2018 international conference on big data and education* (pp. 67–71).

IEEE Robotics and Automation Society. (1984). *ICRA - IEEE Robotics and Automation Society.* Retrieved from https://www.ieee-ras.org/conferences-workshops/fully-sponsored/icra

Kalapos, A., Gór, C., Moni, R., & Harmati, I. (2021). Vision-based reinforcement learning for lane-tracking control. *ACTA IMEKO*, *10*(3), 7–14.

Khadka, S., & Tumer, K. (2018). Evolution-guided policy gradient in reinforcement learning. *Advances in Neural Information Processing Systems*, *31*.

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A. A. A., Yogamani, S., & Pérez, P. (2021). Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 1-18. doi: 10.1109/TITS.2021.3054625

Li, Z., Xu, C., & Zhang, G. (2021). *A deep reinforcement learning approach for traffic signal control optimization.* arXiv. Retrieved from https://arxiv.org/abs/2107.06115 doi: 10.48550/ARXIV.2107.06115

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Neural Information Processing Systems Foundation. (1987). *Neural Information Processing Systems Foundation.* Retrieved from https://nips.cc/

OpenAI. (2016). *Gym: A toolkit for developing and comparing reinforcement learning algorithms.* Retrieved from https://gym.openai.com/docs/#background-why-gym-2016

Paul, S., Kurin, V., & Whiteson, S. (2019). Fast efficient hyperparameter tuning for policy gradient methods. *Advances in Neural Information Processing Systems*, *32*.

Policy, F. A. V. (2016). Accelerating the next revolution in roadway safety.(2016, sep) washington. *DC: National Highway Traffic Safety Administration*.

Proximal Policy Optimization — Spinning Up documentation. (2018). Retrieved from https://spinningup.openai.com/en/latest/algorithms/ppo.html

Retting, R. A., Weinstein, H. B., & Solomon, M. G. (2003). Analysis of motor-vehicle crashes at stop signs in four u.s. cities. *Journal of Safety Research*, *34*(5), 485-489. Retrieved from https://www.sciencedirect.com/science/article/pii/S0022437503000689 doi: https://doi.org/10.1016/j.jsr.2003.05.001

Saavedra-Ruiz, M., Morin, S., & Paull, L. (2022). *Monocular robot navigation with self-supervised pretrained vision transformers.* arXiv. Retrieved from https://arxiv.org/abs/2203.03682 doi: 10.48550/ARXIV.2203.03682

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning* (pp. 1889–1897).

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Singh, S. (2015). *Critical reasons for crashes investigated in the national motor vehicle crash causation survey* (Tech. Rep.).

Stanford Vision Lab, Stanford University, Princeton University. (2020). *ImageNet.* Retrieved from https://image-net.org/

The Duckietown Foundation. (2016). *Duckietown Foundation – Duckietown.* Retrieved from https://www.duckietown.org/about/duckietown-foundation

Yang, S. (2016, 01). *New 'deep learning' technique enables robot mastery of skills via trial and error.* Retrieved from https://news.berkeley.edu/2015/05/21/deep-learning-robot-masters-skills-via-trial-and-error/

Appendix A

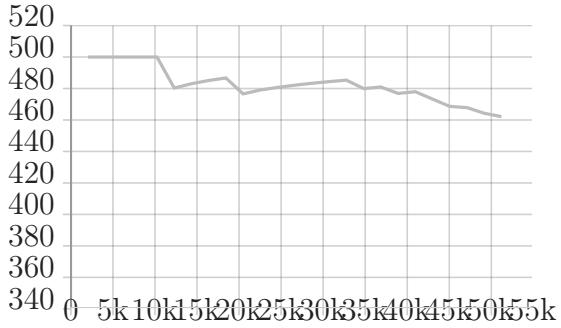
Reward Function Pseudo-code

```
if episode has started and speed > "stop" threshold
    and if current distance to goal is < previous distance to goal
        reward = 1
    else if current distance to goal > previous distance to goal
        reward = -1
else if the episode has started (for at least 10 timesteps) and speed <= "stop" thr
    set done flag
    give final reward of 1
else (meaning episode is at the start frame)
    set distance to goal to a high number
    reward = 0
```

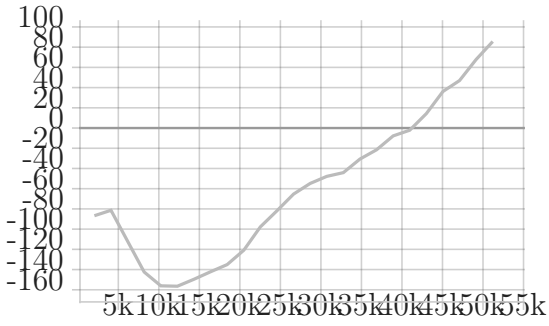
Appendix B

Training Data

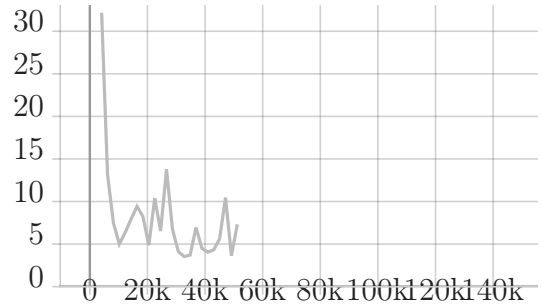
B.1 PPO4



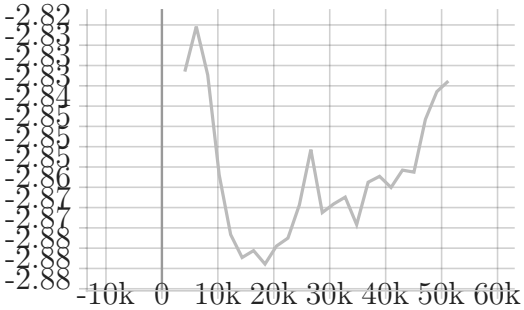
(a) Episode length mean "PPO4"



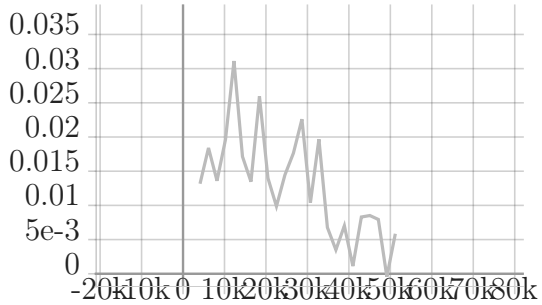
(b) Episode reward mean "PPO4"



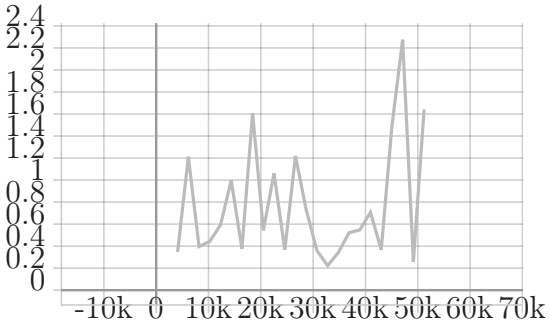
(c) Value loss "PPO4"



(d) Entropy loss "PPO4"

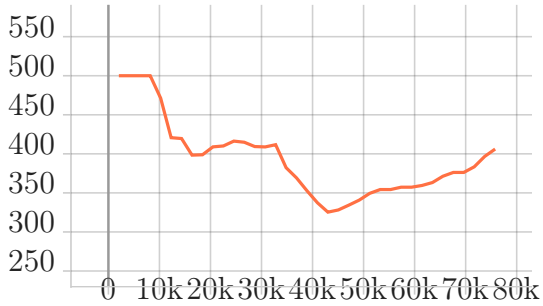


(e) Policy gradient loss "PPO4"

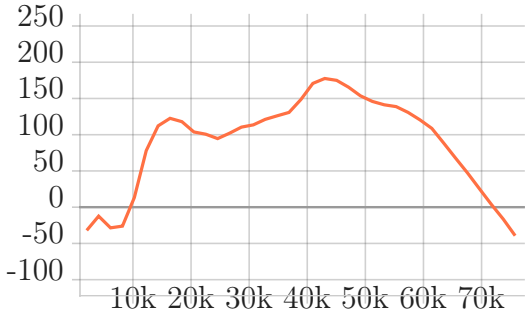


(f) Loss "PPO4"

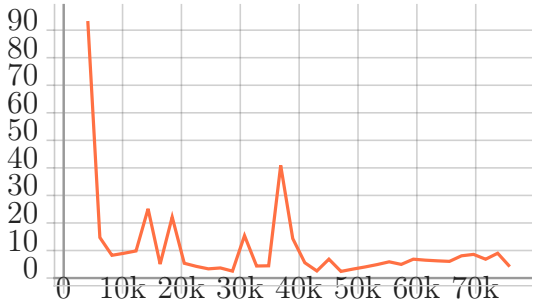
B.2 PPO25



(a) Episode length mean

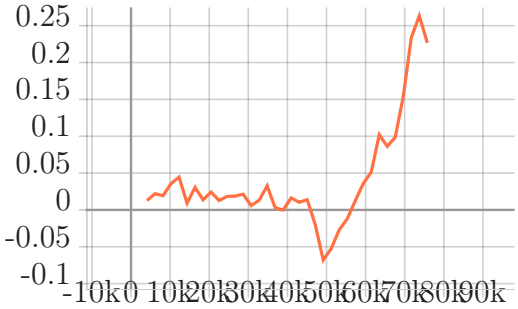


(b) Episode reward mean

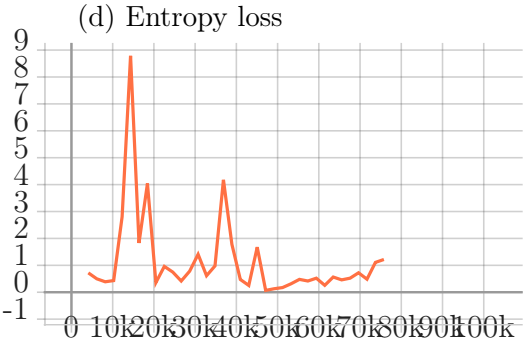


(c) Train value loss

(d) Entropy loss

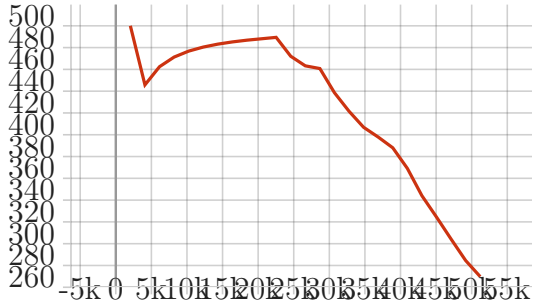


(e) Policy gradient loss

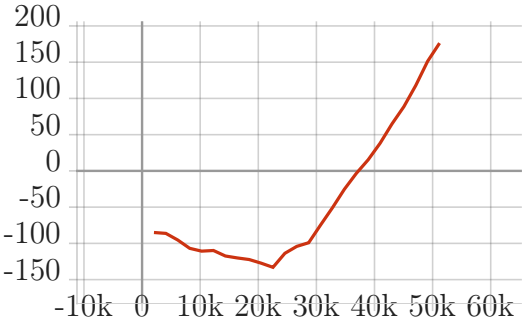


(f) Loss

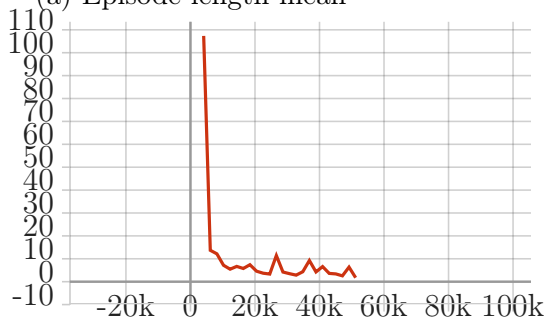
B.3 PPO30



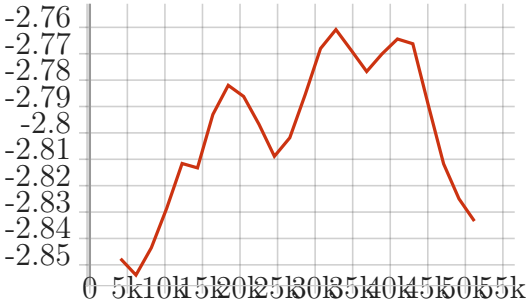
(a) Episode length mean



(b) Episode reward mean



(c) Train value loss

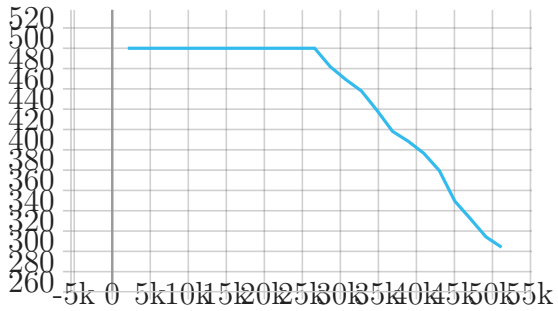


(d) Entropy loss

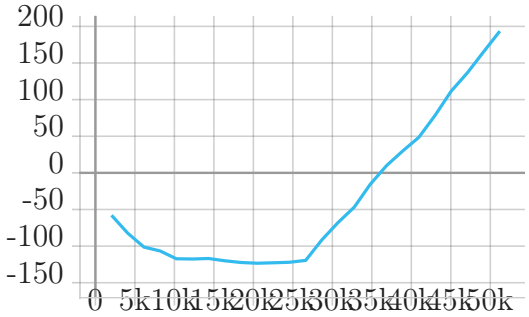
(e) Train value loss

(f) Entropy loss

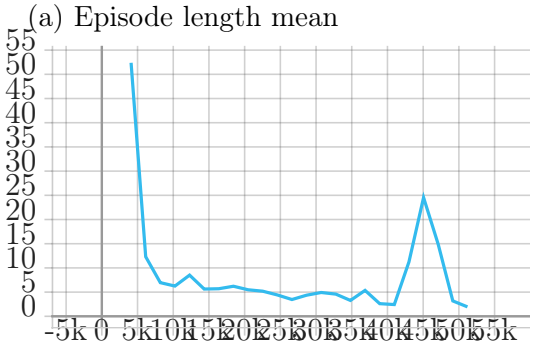
B.4 PPO31



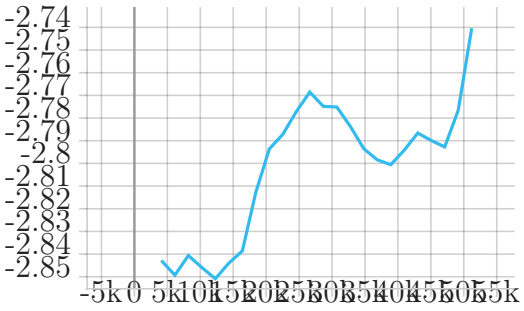
(a) Episode length mean



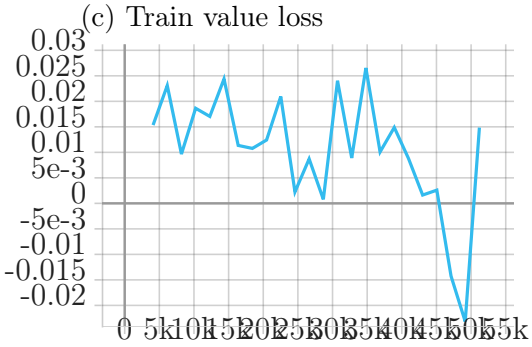
(b) Episode reward mean



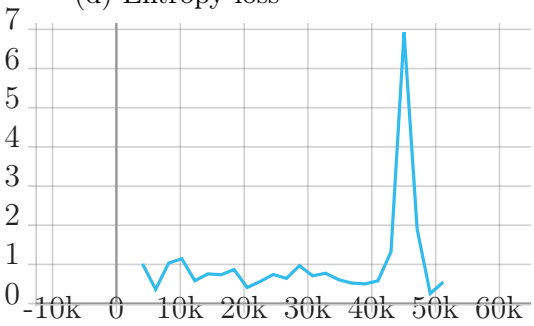
(c) Train value loss



(d) Entropy loss

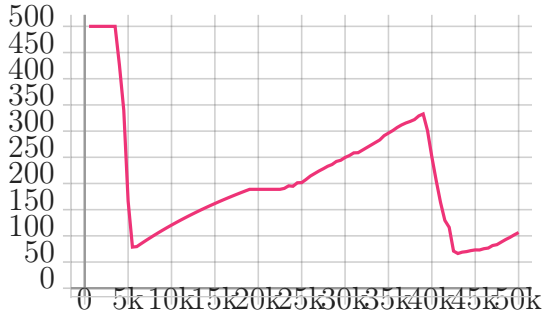


(e) Train value loss

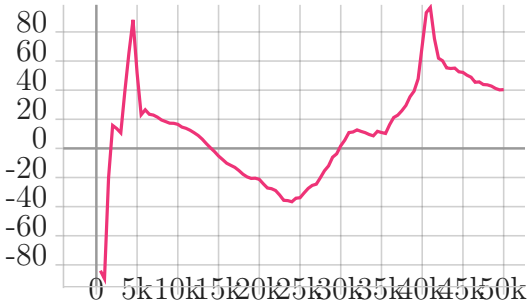


(f) Entropy loss

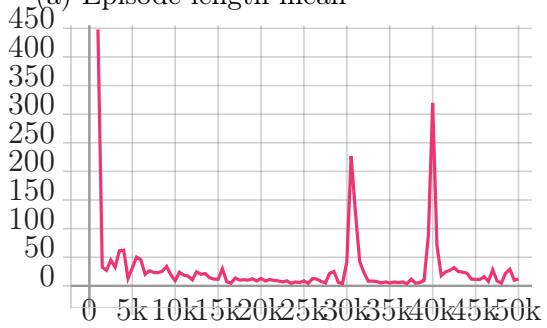
B.5 STEP500



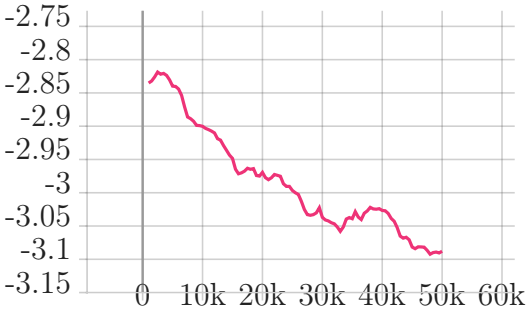
(a) Episode length mean



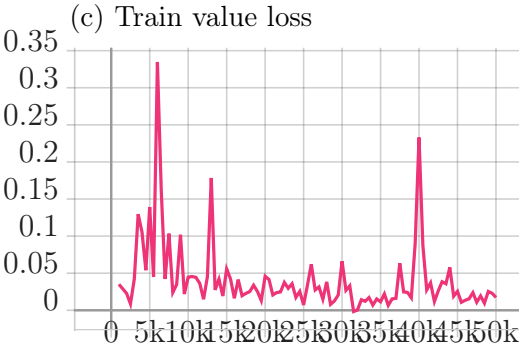
(b) Episode reward mean



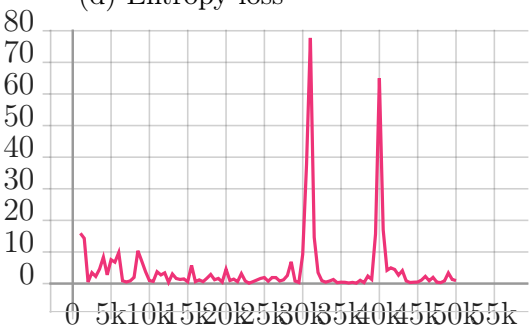
(c) Train value loss



(d) Entropy loss

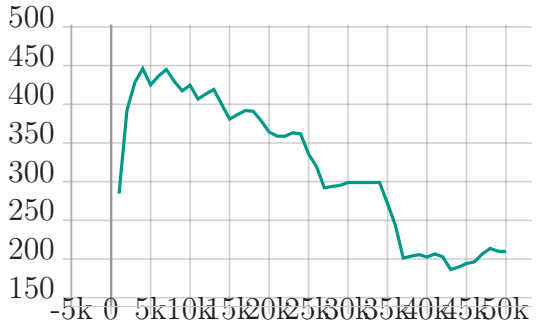


(e) Train value loss

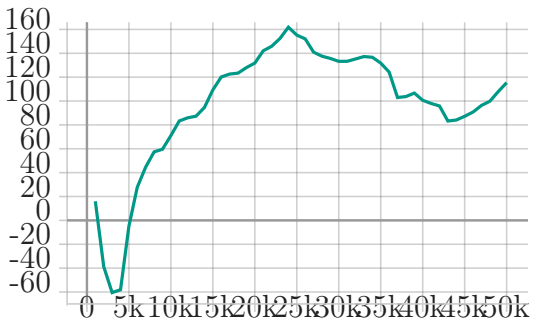


(f) Entropy loss

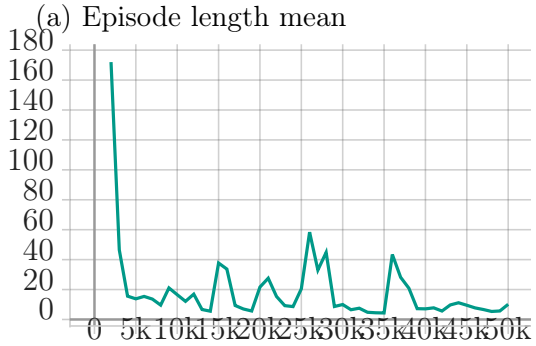
B.6 STEP1000



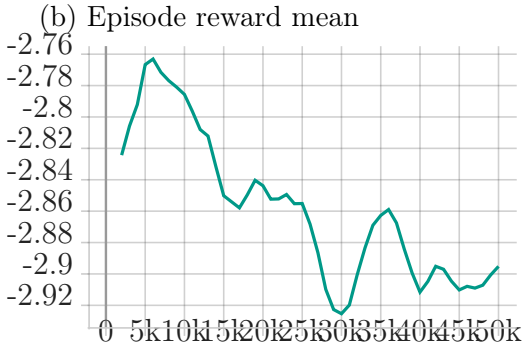
(a) Episode length mean



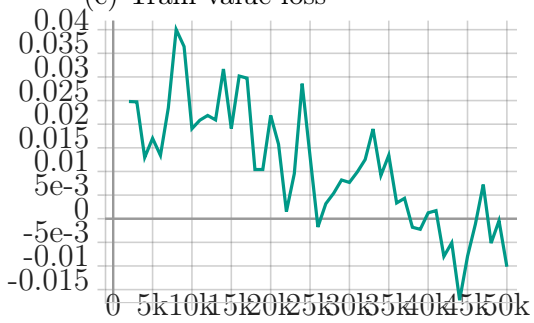
(b) Episode reward mean



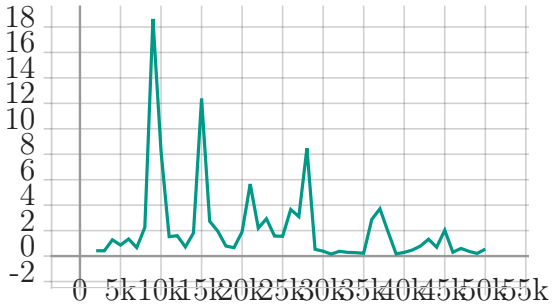
(c) Train value loss



(d) Entropy loss



(e) Train value loss



(f) Entropy loss