UPPSALA
UNIVERSITET

MULTI-AGENT REINFORCEMENT LEARNING WITH APPLICATION ON TRAFFIC FLOW CONTROL

Submitted by
Mikael J. Olsson

A thesis submitted to the Department of Statistics in partial fulfillment of the requirements for a two-year Master of Arts degree in Statistics in the Faculty of Social Sciences

Supervisors
Jieqiang Wei, Ericsson
Ronnie Pingel, Uppsala Universitet

Spring, 2021

ABSTRACT

Traffic congestion diminish driving experience and increases the CO_2 emissions. With the rise of 5G and machine learning, the possibilities to reduce traffic congestion are endless. This thesis aims to study if multi-agent reinforcement learning speed recommendations on a vehicle level can reduce congestion and thus control traffic flow. This is done by simulating a highway with an obstacle on one side of the lanes, forcing all the vehicles to drive on the same lane past the obstacle, resulting in congestion. A game theory aspect of drivers not obeying the speed recommendations was implemented to further simulate real traffic. Three Deep Q-network based models were trained on the highway and the best model was tested. The tests showed that multi-agent reinforcement learning speed recommendations can reduce the congestion, measured in vehicle hours, up to 21% and if 1/3 of the vehicles uses the system, the total congestion can be significantly reduced. In addition, the test showed that the model achieves a success rate of 80%. Two improvements to the success rate would be more training and implementing a non reinforcement learning mechanism for the autonomous driving part.

Key words: *Deep Q-Network, Game Theory, Shielding, Simulations.*

Video example of the results: https://www.youtube.com/watch?v=X8NY9oE5MbY

Contents

Nomenclature

γ	Discount rate
Π	Joint policy
π	Policy (decision-making rule)
π^*	The best policy
ρ	Collaboration metric
ρ	Single-agent reward function coefficient
θ	Weights of a Neural Network when estimating the value function for a Deep-Q Network
A	Set of actions, named action space
a	An action
D	Stored experience
P_a	Probability that action a at time t and state s will lead to a transition to state s
Q_t	Estimated action values at time t
r	An reward
R_a	Expected immediate reward received from transitioning from s to s' under action a
S	Set of states, named state space
s, s'	States
t	Discrete time step
U	Minibatch or sample of experience

1 Background

Traffic congestion is a state in traffic which is defined by decreased speed and queuing. The state occurs when an available road capacity is saturated, i.e. the amount of traffic creates a demand for space greater than the available capacity. Tomtom [1] found that an average person driving to work in Stockholm spends in total four days and two hours in traffic congestion every year. In 2008, Barth and Kanok [2] analyzed the real-world carbon dioxide impacts of traffic congestion and concluded that congestion during peak hour increases the CO_2 emissions by approximately 8%. Thus, reducing traffic congestion would be beneficial in two main ways. Firstly, it would enhance the driver experience by reducing queues. Secondly, it would reduce the CO_2 emissions by creating a smoother traffic flow.

Electronic speed limit signs which varies depending on the traffic has been implemented in many countries. This technique is called *variable speed limit* and is a part of the smart city vision where urban areas use different types of electronic methods and sensors to collect data. In USA, variable data on speed limits has been evaluated and it shows that implementing this technique can generate preferential system benefits in terms of traffic efficiency and safety [3]. However, variable speed limits, as with regular speed limit signs, are fixed to their respective position. With the rise of 5G and machine learning, one could ask themselves if they could be implemented with variable speed limits in vehicle systems, resolving in a machine learning speed recommendation system powered by data from 5G networks.

The aim of this thesis is to study if reinforcement learning controlled speed recommendations could reduce traffic congestion. This will be done by simulating a highway where congestion is created. At each time step the vehicles will get an individual speed recommendation. One can see it as a control tower collecting data and giving the cars speed recommendations through 5G. To actually implement the speed recommendations, the vehicles will be controlled by reinforcement learning which can take the actions accelerate, brake, change lane and idle.

The game theory aspect of drivers obeying the recommendations to a certain level will be explored, together with the ratio of cars with speed recommendation needed to reduce congestion. This will be measured by simulating a highway, with different settings, multiple times and at each setting capture the data of interest.

The remainder of this thesis is structured as follows. Section 2 presents the theory used in this thesis, with the main focus of reinforcement learning. In section 3, the simulation environment and its setup is explained, together with the possible methods. In section 4, the outcome of the training and testing simulation results are analyzed. In section 5, the results are discussed in great detail and the conclusions drawn are presented with the aim of answering the research question rigorously. Lastly, section 6 will summarize the thesis, followed by the references, acknowledgements and an appendix.

1.1 Related work

Several studies on the implementation of reinforcement learning to reduce traffic congestion has been done. Wiering [4] studied the effect of a multi-agent system for traffic lights in four-way intersections, showing that the reinforcement learning systems could outperform the benchmark non-adaptable systems and that value functions can be applied to the driving policies of the cars to find the optimal route. Bakker and Spaan [5] implemented reinforcement learning onto speed limits before an on-ramp (which often reduce traffic flow) to evaluate its performance on reducing congestion, resulting in a model performing well on small road networks. Similar studies show equivalent and agreeing results [6] [7]. The mentioned studies give speed recommendations on fixed occurrences of the road, as speed signs. Therefore, they do not study the effect of reinforcement learning speed recommendations on individual cars anywhere on the road.

Ha *et al.* [8] explored the possibility to improve traffic flow through reinforcement learning controlled autonomous cars. They concluded that in a traffic bottleneck, reinforcement learning controlled autonomous cars could enhance the traffic flow significantly. They also concluded that if only 10% of the cars were autonomously driven, the traffic flow could still be enhanced, albeit by a smaller effect. Since the study focuses on self-driving cars, it does not include the game theory aspect of speed recommendations and whether those are obeyed or not.

2 Theory

This chapter will in detail go through the underlying theory of the methods used. The basics of reinforcement learning will be explained, followed by central aspects of reinforcement learning used in this thesis, e.g. markov decision process, exploration vs exploitation and multi-agent reinforcement learning. To fully grasp the models (Deep Q-network and Double Deep Q-network) used in the simulations, the theory behind temporal-difference learning and Q-learning is explained. In addition, a part will cover essential traffic concepts and the statistical theory and tests used in this thesis.

2.1 Methods for Reinforcement Learning

Reinforcement learning (RL) is one of the three sub-groups of machine learning, supervised- and unsupervised learning being the other two groups. In RL, an agent learns how to optimally act in an environment, based on a trial-and-error behaviour interaction [9]. This process is commonly formulated as a markov decision process (discussed in the next section). While supervised learning use labelled data as input, RL has no labelled data to learn from. RL is instead goal-oriented, meaning that it builds a learning model to find the optimal long-term aim by learning from trial-and-error. An action that leads to an improved outcome is reinforced in the agent's set of behaviours, hence the word *reinforcement*. To measure how good an action is at a certain time step, a reward function is created. This reward function calculates a reward (positive or negative) at each time step. Often immediate reward is more important to the agent and the researcher. Hence, the goal of RL is to maximize the cumulative reward given a predetermined discount rate that describes how important future reward is.

2.1.1 Markov decision process

A Markov Decision Process (MDP) [10] is a form of sequential decision making, where actions effect immediate rewards, subsequent situations (or states) and future rewards. Therefore MDPs considers delayed reward and consequently the tradeoff between immediate and delayed reward. A MDP is defined as

Definition 2.1 (Markov Decision Process)**.** A MDP is a tuple $\langle S, A, P_a, R_a \rangle$, where

- S is the set of states named *state space.*
- A is the set of actions named *action space.*
- $P_a(s, s')$ or $Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ is the probability that action a at time t and state s will lead to a transition to state s'.
- $R_a(s, s')$ is the expected immediate reward received from transitioning from s to s' under action a.

Figure 1 shows agent-environment interaction in a MDP, where the agent receives a reward and a state from the environment. From that information, the agent takes an action, which the environment receives and in return gives the state and reward for the next time step. This process is repeated until the episode reaches a terminal state.

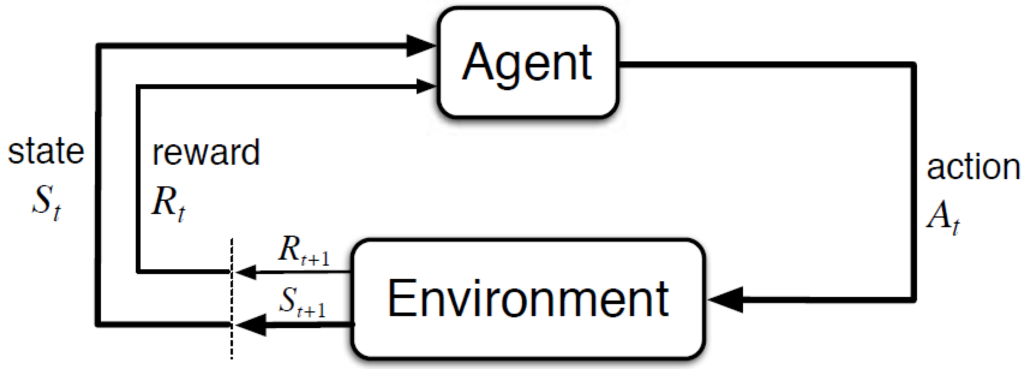

Figure 1: The environment in a Markov decision process [9].

The aim in a MDP is to find the best *policy*, a function π that chooses the action, $\pi(s)$ that should be taken in state s. When a policy is appended to a MDP, which fixes the action for each state, the results is a *Markov chain*. To find the best policy, denoted π^*, the objective is to maximize the sum of discounted cumulative random rewards,

$$\pi^* = \max_{a \in A, s \in S} \left[E\big[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) \big] \right], \tag{1}$$

where $\gamma \in [0, 1)$ is the discount rate. A lower discount factor incentives the agent to favor taking actions that accumulate rewards early, rather than late. A high discount rate informs the agent that future rewards are almost as important as present rewards.

2.1.2 Exploration vs Exploitation

One of the biggest challenges in reinforcement learning is the trade-off between exploration and exploitation. To obtain a high reward, the agent should take actions with the highest estimated reward (*exploit*), which is often learned by passed experiences. Nevertheless, to obtain that experience, the agents has to *explore* actions it has not selected before. The caveat is that neither exploration nor exploitation can be used exclusively without failing at the other. This is called the exploration vs exploitation dilemma. On a stochastic mission, every action should be tested extensively to obtain a reliable estimate of the corresponding expected reward. This dilemma has been studied immensely for decades without being resolved [11].

One popular approach is the ϵ-*greedy*. In this approach the agent takes the action with the highest estimated value most of the times. This action is a *greedy* action and is defined as

$$A_t \doteq \operatorname*{argmax}_a Q_t(a), \tag{2}$$

where $Q_t(a)$ is the estimated action values at time t, i.e. the expected return from taking action a in a certain state s at time t. As aforementioned, the ϵ-greedy behaves greedily most of the time. However, every once in a while, with small probability denoted ϵ, the agent selects an action randomly from all the actions, with equal probability. This assures that $Q_t(a)$ will in the limit converge to the true action values. As the number of steps increases, all actions will be sampled an infinite number of times. Consequently, selecting the optimal action converges to greater than $1 - \epsilon$, which is, to near certainty. Now consider the implementation of ϵ-greedy:

Algorithm 1: ϵ-greedy implementation

```
Data: Q: Q-table generated so far, $\epsilon$: a small number, S: current state
Function SELECT-ACTION(Q,S,$\epsilon$) is
    $n \leftarrow$ uniform random between 0 and 1;
    if $n < \epsilon$ then
        A $\leftarrow$ random action from the action space;
    else
        A $\leftarrow$ maxQ(S,.);
    end
    Return selected action A;
end
```

Worth mentioning is that there are countless of approaches to the exploration vs exploitation dilemma, like the *epsilon-first strategy* and the *epsilon-decreasing strategy*, to name a few [9]. However, ϵ-greedy is used in this thesis since it is the widest used in the research field.

2.1.3 Multi-agent Reinforcement Learning

Up to now, the focus has been on single-agent RL, where the framework only consider a maximum of one agent. The focus will now shift to a multi-agent focus, with at least two or more agents. As in the single-agent setting, the agents still learns through trial-and-error. Nevertheless, the main difference is that the reward function and state spaces are influenced by the joint actions of all agents. Resulting in agents needing to not only consider the environment but the other agents as well. Thus the MDP for a multi-agent reinforcement learning (MARL) problem is generalized. This generalization of the MDP is the stochastic game [12], defined as

Definition 2.2 (Stochastic game)**.** A stochastic game is a tuple $\langle S, A_1, ..., A_n, f, \rho_1, ..., \rho_n \rangle$, where n is the number of agents, S is the finite set of states, A_i, where $i = 1, .., n$, is the finite available actions for the corresponding agent and ρ_i, $i = 1, .., n$ is a collaboration metric. This results in the joint action set $\mathbf{A}$ equal to $A_1 \times \cdots \times A_n$, the state probability function is therefore $f{:}S \times \mathbf{A} \times S \rightarrow [0, 1]$ and $\rho_i{:}S \times \mathbf{A} \times S \rightarrow \mathbb{R}, i = 1, .., n$, are the reward functions for the agents.

Assuming bounded reward functions, in the MARL setting, state transitions are the result of the joint action, $\mathbf{a_t} = [a_{1,t}^T, ..., a_{n,t}^T]^T$, $\mathbf{a_t} \in \mathbf{A}$, $a_{i,t} \in A_i$. Thereafter, the policies $\pi_i{:}S \times A_i \rightarrow [0, 1]$ form $\mathbf{\Pi}$, the joint policy. Since the agents reward depends on the joint action, their expected return depends on $\mathbf{\Pi}$, where

$$R_i^{\mathbf{\Pi}}(x) = E\left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) \mid x_0 = x, \mathbf{\Pi}\right]. \tag{3}$$

The goal for each agent can differ from a fully cooperative goal to a fully competitive. Under the fully cooperative goal, the reward function for each agent is the same: $\rho_1 = \cdots = \rho_n$, where ρ is the single-agent goal. On that account, the agents try to maximize the same goal. On the contrary, if $n = 2$ and $\rho_1 = -\rho_2$, the stochastic game is fully competitive. Fully competitive stochastic games can occur when there is more than three agents. Nevertheless, in research two agents are standard [12]. In this thesis, the stochastic game is fully cooperative, since the goal for all agents maximize the traffic flow.

In MARL there are different types of learning paradigms. Yang and Wang [13] illustrates these paradigms in Figure 2. In the first paradigm, the agents share policy but takes action independently from each other. In the second paradigm, the agents' actions and policies are independent. In the third paradigm, the agents share policies in groups, with independent actions. In the fourth paradigm, one central controller controls all: agents can share information with each other at any time point. In the fifth paradigm, during training the agents can share information and during testing they can not. Lastly, in the sixth paradigm, in training the agents share information to their neighbours and during testing they do not.

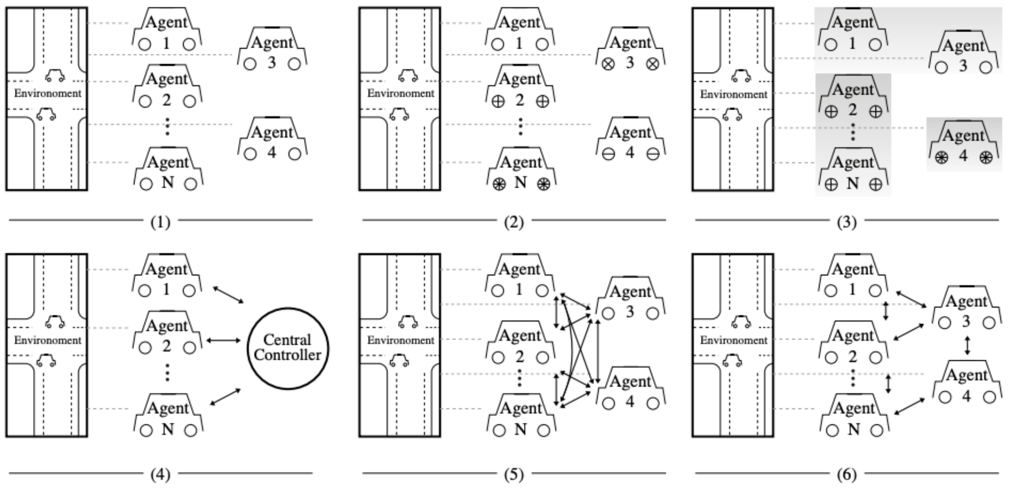

Figure 2: Common Learning paradigms in MARL Algorithms [13].

2.1.4 Shielding

In 2018, Alshiekh *et al.* [14] introduced a new method, named *shielding*, for models to learn optimal policies while enforcing properties in form of temporal logic. Given a temporal logic specification that is forced to be obeyed by the agents, a shield is proposed. If the agents take the wrong action, the shield corrects the action. Shielding is a part of *safe reinforcement learning* and has in tests showed that it can be more data efficient, i.e. finding the optimal policy faster. Instead of taking actions leading to termination, the agents are corrected. Thus, the exploration per episode will be bigger. Leading to more efficient and safer learning [14]. Therefore, shielding will be used in this thesis.

An example of shielding is in a game of Texas Hold'em, where the agent wants to go "all-in" (wager all their chips) on a 2-7 offsuit hand, the sheilding can prevent that action and choose a more sensible action. This would be smart since a 2-7 offsuit hand is one of the worst hands with a 4.8% winning probability on a table of 10 players, leading to a 95.2% probability of termination when going "all-in".

2.1.5 Temporal-Difference Learning

Temporal-Difference learning (TD) is a learning technique with both Monte Carlo approaches and dynamic programming approaches. As for Monte Carlo methods, TD methods learn from raw experience, i.e. without a model of the environment's dynamics. As for dynamic programming methods, TD updates estimates based in part on other learned estimates, by bootstrapping.

TD methods use past experience to predict the future. Given experience from policy π, the method updates its estimate V of v_π from the nonterminal states S_t occurring in that experience. TD methods update the estimated state value time t and $V(S_t)$, at time step $t+1$, using the observed reward R_{t+1} and the estimated $V(S_{t+1})$. The simplest TD method makes the update

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\Big], \tag{4}$$

directly after the transition to S_{t+1} and after receiving R_{t+1}. This method is called $TD(0)$ [9].

2.1.6 Q-learning

In 1989, Watkins [15] proposed an off-policy temporal difference algorithm famously known as *Q-learning.* An off-policy algorithm uses two policies; *target-* and *behavior* policy. The policy being studied is the target policy and the behavior policy is used to explore and generate behavior. Therefore, the policy learns from data "off" the target policy[1]. The Q-learning algorithm is defined as

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_{t+1}, A_t)\Big]. \tag{5}$$

The optimal action-value function, q_*, is estimated by the learned action value function, Q. The weight coefficient is denoted α .The optimal action-value function is estimated inde-

[1]Opposite to the off-policy is the on-policy, which learns from data "on" policy. Thus, it learns action values for a near-optimal policy, since it still explores.

pendently of the policy being followed. This has been shown to converge, if the assumption that all state-action pairs are continuously updated, is fulfilled. Q will converge to q_* with probability 1, under the previous assumption together with a version of the stochastic approximation conditions on the course of step-size parameters. The implementation of the Q-learning algorithm is shown below.

Algorithm 2: Q-learning for estimating the best policy

```
Data: $\alpha$: step size, : a small number
Initialize $Q(s,a)$, for all $s \in S^+$, $a \in A(s)$, arbitrarily except that $Q(terminal, .) = 0$
for each episode do
    Initialize $S$;
    for each step of the episode do
        Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\epsilon$-greedy);
        Take action $A$, observe $R$, $S^{'}$;
        $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S^{'}, a) - Q(S,A)\big]$;
        $S \leftarrow S^{'}$;
    end
    Until $S$ is terminal;
end
```

Although Q-learning is a popular reinforcement learning algorithm, it has been shown to perform poorly in stochastic MDPs due to overestimation of the action values (see appendix for derivation). These overestimations come from a positive bias due to the fact that Q-learning approximates the maximum expected action value with the maximum action value.

Hasselt [16] proposed a different way to approximate the maximum expected value. This is the *double Q-learning.* The Q-learning estimates the value for the next state, $\max_a Q_t(s_{t+1}, a)$, with $E[\max_a Q_t(s_t, a)]$, resulting in a single Q function. As the name suggests, the double Q-learning algorithm uses two Q functions: Q^a and Q^b. Both Q functions are updated for the next state with estimates from each other. On each step, one of the Q functions are randomly chosen to be updated using the other, resulting in the following expression if Q^a was chosen

$$Q^a(S_t, A_t) \leftarrow Q^a(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma Q^b\big(S_{t+1}, \underset{a}{\operatorname{argmax}}\, Q^a(S_{t+1}, a)\big) - Q^a(S_t, A_t)\Big]. \tag{6}$$

Otherwise, the Q^a and Q^b would switch places in equation 6, so that Q^b is updated. This solution has been shown to reduce the positive bias [16]. Nonetheless, it is crucial that the Q-functions learn from different data. However, when selecting an action both Q-functions should be used to not be less data-efficient than Q-leaning.

2.1.7 Deep Q-Learning

Remember that the common version of Q-learning has a lookup table of values with one row for each state-action pair. So to learn the optimal Q-value function, the Q-learning algorithm makes use of the Bellman equation (equation 5) for the Q-value function. Instead of a lookup table the *Deep Q-Network* (DQN), presented by Mnih *et al* [17], paramaterize an estimated value function $Q(s, a; \theta_t)$ using a neural network as for example a convolutional neural network (further explanation of neural networks are considered out of scope in this thesis, since the focus is reinforcement learning), where θ_t represent the weights of the network at time step t. The function estimator is trained by minimizing the following loss at each time step t and is determined as

$$L(\theta_t) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\big[(y_t^{DQN} - Q_t(s, a; \theta_t))^2\big], \tag{7}$$

where

$$y_t^{DQN} \equiv R_{t+1} + \gamma \max_a Q(s_{t+1}, a; \theta_t). \tag{8}$$

and D is the stored experience. During training, Q-learning updates are applied on mini-batches or samples of experience $(s, a, r, s') \sim U(D)$, drawn uniformly at random from the stored experience. y_t^{DQN} is called the target of the DQN. As discussed in the Q-learning section the *max* operator in the target makes it more likely to overestimate values. Hence, the double Q-learning was introduced. A neural network can be applied into a double Q-learning network. Creating the *double deep Q-network* [18] (DDQN). The target, y_t^{DQN} in a DDQN is defined as

$$y_t^{DDQN} \equiv R_{t+1} + \gamma \max_a Q\big(s_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a; \theta_t); \theta_t^-\big), \tag{9}$$

where θ_t^- and θ_t are two different sets of experience.

DQN and DDQN will be the models used in the simulations. This is because of them being available to use in a MARL setting when simulating. Other models, as the *Distributed-Q*, would be equally good contenders if they were available.

2.2 Methods for Analyzing the Results

This section will explain the methods that will be used to analyze the results. First, the measurement for congestion will be defined. Second, the tests that statistically determine the effects will be presented.

2.2.1 Vehicle Hours

Vehicle hours [19] is a metric that capture the relationship between congestion and the travel time of vehicles. One vehicle hour correspond to 1 vehicle driving on a road for 1 hour, yet it can also correspond to 60 vehicles driving on the same road for 1 minute. So by aggregating over all vehicles, the vehicle hours are captured. In this thesis the mean vehicle hour will be used, since the number of cars will be drawn uniformly at random from a predetermined range.

Vehicle hours was chosen as the metric to capture congestion due to three reasons. Firstly, vehicle hours is a highly used metric in the research field. Secondly, it is easy to interpret. Thirdly, vehicle hours is not too computationally heavy to capture and calculate during simulation.

Other metrics, as Fundamental Diagram [20], would probably be equally good or better to capture congestion. However, with the cost of computational power and/or a need for real traffic data. They can therefore not be implemented in this thesis.

2.2.2 Two-sample Kolmogorov–Smirnov Test

The two-sample Kolmogorov–Smirnov test [21] is used to test whether two probability distributions are the same or not. The null hypothesis assumes that the probability distributions do not differ. The statistic is determined as

$$D_{n,m} = \sup_x \mid F_{1,n}(x) - F_{2,m} \mid, \tag{10}$$

where $F_{1,n}$ and $F_{2,m}$ are the empirical distribution functions sampled respectively. For large sample sizes, the null is rejected at significance level α if

$$D_{n,m} > c(\alpha)\sqrt{\frac{n+m}{n \times m}}, \tag{11}$$

where n and m are the sample sizes and $c(\alpha)$ is generally from

$$c(\alpha) = \sqrt{-\ln\left(\frac{\alpha}{2}\right) \times \frac{1}{2}}, \tag{12}$$

where the null is that the samples are drawn from the same distribution. The alternative hypothesis is that the samples are not drawn form the same distribution. It is good to point out that the test only tests if the data is from the same distribution, and not which distribution is from.

2.2.3 One-way ANOVA, Welch's ANOVA & Post-hoc Tests

One-way ANOVA compares the means between a group of samples and statistically determines whether any of those significantly differ from each other or not [22]. More exactly, it tests the null hypothesis

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_k, \tag{13}$$

where μ is the sample mean and k is the number of samples. Intuitively, the alternative is that at least one mean or more differs. To perform the one-way ANOVA, three assumptions should be fulfilled. The first assumption is that the distribution of the residuals are normal (where the residuals usually come from a linear regression on the dependent variable). The second assumption is that there is homogeneity of variances, i.e. homoscedasticity. The third assumption is that the samples are independently sampled, each sample is randomly selected and independent.

Consider the following sum of squares

$$SST \equiv \sum_{i=1}^{k}\sum_{j=1}^{n}(y_{ij} - \bar{\bar{y}}) = \sum_{i=1}^{k}\sum_{j=1}^{n} y_{ij}^2 - \frac{(\sum_{i=1}^{k}\sum_{j=1}^{n} y_{ij})^2}{kn} \tag{14}$$

$$SSA \equiv \frac{1}{n}\sum_{i=1}^{k}\left(\sum_{j=1}^{n} y_{ij}\right)^2 - \frac{1}{kn}\left(\sum_{i=1}^{k}\sum_{j=1}^{n} y_{ij}\right)^2 \tag{15}$$

$$SSE \equiv \sum_{i=1}^{k}\sum_{j=1}^{n}(y_{ij} - \bar{y_i})^2 = SST - SSA, \tag{16}$$

which are the total, treatment and error sums of squares. There $\bar{y_i}$ is the mean of sample group i, $i = 1, 2, ..., k$, n is the number of observation in a group and $\bar{\bar{y}}$ is the mean of all samples, or mean of means if you like. Now, one can calculate the tests corresponding F-score by $F_{ANOVA} = MSA/MSE \sim F_{k,n-k}$, where $MSA = SSA/(k-1)$ and $MSE = SSE/(kn-1)$.

The assumption of homoscedasticity is a strong assumption that is often hard to fulfill. If that is the case, one should use the *Welch's ANOVA* [23]. The Welch's ANOVA is similar to the one-way ANOVA, they have the same hypothesis and assumptions, expect for the homoscedasticity assumption. Homoscedasticity is not an assumption for the Welch's ANOVA. Now, consider the following treatment sum of squares and weighted mean

$$SST_{welch} \equiv \sum_{i=1}^{k} w_i(\bar{y_i} - \bar{y}_{welch}) \tag{17}$$

$$\bar{y}_{welch} = \frac{\sum_{j=1}^{k} w_j \bar{y_j}}{\sum_{j=1}^{k} w_j}, \tag{18}$$

where w is weights to reduce the variance, calculated as $w_j = n_j/s_j^2$. The s_j term is the observed variance for group j. Now, the F-score is calculated as

$$F_{welch} = \frac{SST_{welch}/(k-1)}{1 + \frac{2\Delta(k-2)}{3}} \sim F_{k-1,1/\Delta}, \tag{19}$$

where

$$\Delta = \frac{3\sum_{j=1}^{k} \frac{\left(1 - \frac{w_j}{\sum_{j=1}^{k} w_j}\right)^2}{n_i - 1}}{k^2 - 1}. \tag{20}$$

Post-hoc tests are a vital part of ANOVA testing. Whilst ANOVA statistically tests if at least one mean in a group differs, it does not identify which particular mean, or means, that differs. Therefore, one can use post-hoc tests to explore the differences between multiple group means while controlling the error rate (experiment-wise). There exists a few post-hoc tests, this thesis will focus on the *Games-Howell post-hoc test* [24] since it does not assume homoscedasticity. The Games-Howell post-hoc test is a nonparametric approach to compare combinations of groups built from the Welch's degrees of freedom correction. The assumptions are the same as for the Welch's ANOVA. The test is defined as

$$\bar{x_i} - \bar{x_j} > q_{\sigma,k,df}, \tag{21}$$

where

$$\sigma = \sqrt{\frac{1}{2}\left(\frac{s_i^2}{n_i} + \frac{s_j^2}{n_j}\right)} \tag{22}$$

$$df = \frac{\left(\frac{s_i^2}{n_i} + \frac{s_j^2}{n_j}\right)^2}{\frac{\left(\frac{s_i^2}{n_i}\right)^2}{n_i - 1} + \frac{\left(\frac{s_j^2}{n_j}\right)^2}{n_j - 1}}. \tag{23}$$

and k is the number of groups. Lastly the p-values are calculated using Tukey's studentized range, $q_{t\times\sqrt{2},k,df}$, where t is the t-value and is found with Welch's t-test.

3 Simulation Environment Setup

This section will in detail go through the structure and assumptions of the simulations in this thesis. The simulation environment used and the configurations of it will first be explained, followed by the reward function and observation space. Then, the potential models are discussed.

3.1 Environment

The simulations are performed in the environment highway-env [25] through Python. The environment can simulate different traffic situations such as a highway, roundabout and more. In these situations the cars could either be agents or considered to be driven by humans. The human driven simulated vehicles follow a simple but realistic behavior that determines how they accelerate and steer on the road. This behaviour is captured by the Intelligent Driver Model (IDM) by Treiber, Hennecke and Helbing [26]. The IDM model will be used as a benchmark to compare the RL algorithms with, since it mimics human drivers.

A time step is set to be a .5 second jump, meaning that at every half a second, all the agents observe their environment, preform an action and the model thereafter calculates the reward. A natural trade-off between precision and computational cost appears when setting the time step. In this case, a speed recommendation every half a second should be frequent enough to be able to reduce the congestion. We tested the .5 time step against a time step of 1 second and it showed remarkably better performance in the terms of highest yielded reward and fastest learning. Intuitively, if this system was tested in reality, a driver that gets speed recommendations more often than each .5 second could easily be distracted and become a danger in the traffic.

The multi-agent learning paradigm available in the environment is the first paradigm described in 2.1.3 and Figure 2. As a reminder, the first paradigm is that the agents share policy but take actions independently from each other. This also leads to the reward per training episode being the average cumulative reward of the agents. This could potentially lead to slower learning than for example paradigm 6. Nonetheless, it will in the limit converge to the optimal policy for each agent [13].

Configurations of the environment had to be done in order to be able to simulate the problem properly. Firstly, an implementation of how much the speed recommendations are obeyed was created. This is to include the game theory aspect of drivers not obeying speed recommendations or rules in traffic. OECD [27] showed in 2006 that the average speed on a road is 10 km/h faster than recommended speed. About 10-20% of drivers drive even faster. Therefore, the cars with speed recommendations will be driving $\mathrm{N}(1.1, 0.05^2)$ times the recommended speed. This means that the cars will be driving about 10% faster than recommended, in average, with a standard deviation of 5%. The standard deviation of 5% was chosen to mimic the difference between how fast drivers drive. This Gaussian noise is generated for every car in the beginning of each simulation and accordingly fixed for each car and simulation.

Secondly, to create a congestion, a specific course was created. This course mimics a two-lane highway. The cars spawn in the beginning of the course with 15 meter space between each other. The congestion is created by an obstacle on either side of the road, forcing all cars to drive in the same lane pass that object. The obstacle spawns at meter 500 with equal probability to spawn at either lane. Note that if an obstacle is spawned on the first lane, a new obstacle cannot spawn on the second lane, because that would lead to the objects fully blocking the cars to complete the course.

Figure 3 shows an example of how the environment visualizes the simulations. The green cars are the agents, i.e. the cars with speed recommendations. The blue cars are the cars without speed recommendations. The yellow squares are obstacles and the squares at the bottom represent the actions and their current estimated value for one random agent. The color scheme goes from light blue to dark red, where light blue represent an action with high reward and dark red represent an action with low reward.

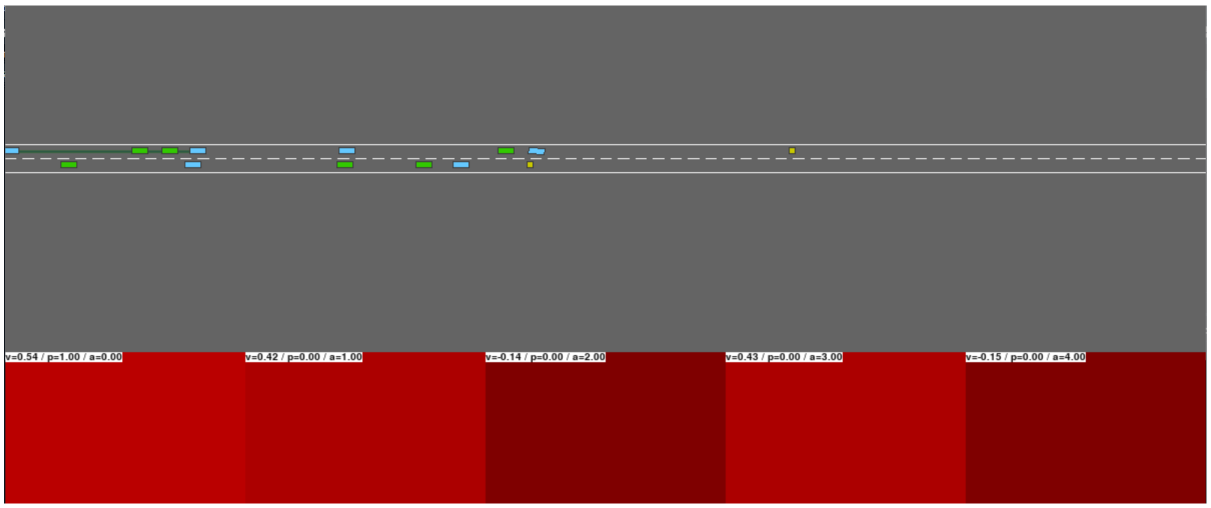

Figure 3: Example of an episode of Highway-env.

For a faster learning, a type of shielding was implemented. This shielding takes form in an auto break if the agent is about to crash. At each time step the IDM model calculates a recommended acceleration. This equation can be found in the appendix. If the recommended acceleration is below a certain threshold, in our case -7, the auto brake is activated and the agent breaks with the recommended deceleration. The shielding takes the following form

$$A_t \leftarrow \begin{cases} \operatorname{argmax}_a Q(s_t, a_t) & \text{if } \phi_t > -7 \text{ and } n_t \leq \epsilon \\ a_t \quad \text{random action} & \text{if } \phi_t > -7 \text{ and } n_t > \epsilon \\ \phi_t & \text{otherwise} \end{cases}, \tag{24}$$

where ϕ_t is the recommended acceleration/deceleration at time step t and n is a uniformly random drawn number from the boundaries [0,1] at time step t, $n_t \in Uniform(0,1)$.

3.2 Reward

Remember that the goal of reinforcement learning is to maximize the reward given the discount rate. Designing a well suited reward function is vital. The function is engineered by the researcher and is often optimized by trial-and-error (just as in RL). However, even though there is no "one solution fits all", there are some guidelines that makes the learning faster. An example is to always normalize the coefficient and the sum of the coefficients [17].

As mentioned in Section 2.1.1, the reward is calculated at each time step for each agent. When an episode is terminated, the average cumulative reward, in the multi-agent setting, is calculated and the policy is updated. The reasoning for the variables used in our reward function is:

- C : The worst case is when an agent crashes and should thus yield the biggest penalty.
- s_s : The agent should not drive too fast nor too slow.
- n_s : The agent should not drive to close to the other cars, causing a dangerous situation.
- c_s : Every agent should take in consider the congestion on the course. Therefore an altruistic penalty that measures the total congestion is needed. Measured as the ratio of vehicles 50 meters behind an obstacle.
- l : Since obstacle in the course will be blocking lanes at certain times, an incentive to switch lane could nudge the agents to switch lane when needed.
- r : Driving on the right side gives a reward because in real traffic most cars drives on the right side when not overtaking another car.

These variables are used in the following reward function

$$R(s,a) = \begin{cases} f\left(\frac{r(\mathsf{s}_s^q,\mathsf{s}_s^s,\mathsf{n}_s,\mathsf{c}_s\mathsf{l},\mathsf{r}) - r_{min}(\mathsf{s}_s^q,\mathsf{s}_s^s,\mathsf{n}_s,\mathsf{c}_s\mathsf{l},\mathsf{r})}{r_{max}(\mathsf{s}_s^q,\mathsf{s}_s^s,\mathsf{n}_s,\mathsf{c}_s\mathsf{l},\mathsf{r}) - r_{min}(\mathsf{s}_s^q,\mathsf{s}_s^s,\mathsf{n}_s,\mathsf{c}_s\mathsf{l},\mathsf{r})}\right) & \text{if } \mathsf{C} = 0 \\ -20 & \text{if } \mathsf{C} = 1 \end{cases}, \tag{25}$$

where $\mathsf{C} = 1$ means that the agent has crashed and zero otherwise, $f(\cdot)$ linearly map the reward to be bounded between $[-1, 1]$, $r(\mathsf{s}_s^q, \mathsf{s}_s^s, \mathsf{n}_s, \mathsf{c}_s\mathsf{l}, \mathsf{r})$ is the reward when the agent has not crashed, $r_{max}(\mathsf{s}_s^q, \mathsf{s}_s^s, \mathsf{n}_s, \mathsf{c}_s\mathsf{l}, \mathsf{r})$ is the maximum reward possible to receive and $r_{min}(\mathsf{s}_s^q, \mathsf{s}_s^s, \mathsf{n}_s, \mathsf{c}_s\mathsf{l}, \mathsf{r})$ is the minimum when the agent has not crashed. The function $r(\mathsf{s}_s^q, \mathsf{s}_s^s, \mathsf{n}_s, \mathsf{c}_s\mathsf{l}, \mathsf{r})$ is defined as

$$r_s(\mathsf{s}_s^q, \mathsf{s}_s^s, \mathsf{n}_s, \mathsf{c}_s\mathsf{l}, \mathsf{r}) = 0.2\big((1 - \mathsf{s}_s^q) - \mathsf{s}_s^s\big) - 0.15(1 - \mathsf{n}_s) - 0.1\mathsf{c}_s + 0.02\mathsf{l} + 0.03\mathsf{r}, \tag{26}$$

where s_s^q is the scaled speed when driving fast, s_s^s is the scaled speed when driving slow, n_s is the scaled distance to the nearest car or object, c_s is the scaled total congestion on the course, l is an indicator function showing if the car is changing lane and r is an indicator function showing if the car is driving on the right lane.

The scaled variables, denoted with a small s, are all scaled through the equation $\frac{x-x_{min}}{x_{max}-x_{min}}$ where x is the variable of interest and x_{min} and x_{max} are the boundaries of the variable. However, x could exceed these boundaries. For example, n measures the distance between vehicles and the upper boundary is $\mathsf{n}_{max} = 15$. Nonetheless, the distance between two vehicles could be $\mathsf{n}_s = 1000$. If that is the case, then the result of the equation is set to the nearest integer of 1 and 0. In this example it would be set to 1.

Thereafter the result is linearly mapped to a decided range to help the learning. n_s and c_s are linearly mapped to $[0, 1]$. s_s is linearly mapped to $[-1, 1]$. This is because if the agent drives too fast or slow a penalty should be given and if the agent drives at a reasonable speed the agent should get a reward. The boundaries for each variable are explained in the appendix.

3.3 Observation Space & Actions

An observation space, often in the form of a matrix, is what the agents can "see" before they take an action. The matrix should contain the information needed for the agent to take an action that leads to high reward. The observation space used in this simulation study is named *kinematics* and is built-in the highway-env [25]. The kinematics observation is a $V \times F$ array that narrate a list of V nearby vehicles by F determined features. In our case $F = 5$ with features denoted $Vehicle, x, y, v_x, v_y$, where $Vehicle$ is the name of the car, x is the longitudinal position, y is the latitudinal position, v_x is the longitudinal speed and v_y is the latitudinal speed. The values of the features could also be normalized within a fixed range. When the features are normalized, they are relative to the vehicle that will make an action. This vehicle is always described in the first row of the observation space. Table 1 shows an example of a normalized observation space. If fewer vehicles than V (number of rows) are observed, the last rows are placeholders filled with zeros. Other observation spaces could be used here, the kinematics was chosen since it is built-in and the most realistic if

Table 1: Observation Space - An Example

Vehicle	x	y	v_x	v_y
Ego-vehicle	0.05	0.04	0.75	0
Vehicle 1	-0.1	0.04	0.6	0
Vehicle 2	0.13	0.08	0.675	0
...	...	...	...	...
Vehicle V	0.222	0.105	0.9	0.025

this would be implemented in real traffic.

An action space is an array or a matrix with the actions that an agent can take at each time step. The actions space used is an 5×1 array with the following actions:

- **Faster**: Drive the vehicle 5 km/h faster.
- **Slower**: Drive the vehicle 5 km/h slower.
- **Change Lane Left**: Change to the left lane.
- **Change Lane Right**: Change to the right lane.
- **Idle**: Continue on the same lane with the same speed.

These five actions should be enough to control the vehicle.

3.4 Models

Three models are considered in this thesis, all from the python package `rl-agents` [28] on Github. One deep Q-network and two double deep Q-network with different layers[2]. They were chosen since they could easily be applied on the environment in a MARL setting. The models are:

[2]See Appendix for an explanation of the different layers.

- **DQN**: A Deep Q-Network using a Multilayer Percepton with a [256,256] layer to estimate the value function.
- **DDQN**: A Double Deep Q-Network using the same Multilayer Percepton model as the DQN.
- **EGO-DDQN**: A Double Deep Q-Network using Multilayer Percepton with two [64,64] layers and one [64] Attention layer to estimate the value function.

All three models follow the same setup. They all train for 10 000 episodes, where the maximum number of time steps is 100, yielding a maximum possible score per episode of 100. They all follow an ϵ-greedy strategy when training. The ϵ-greedy strategy have a decreasing ϵ depending on the current episode. The reason for having a decreasing ϵ is that exploring is more important in the beginning of the training. Formally the ϵ is set as

$$\epsilon = 0.05 + (1 - 0.05) \times e^{\frac{-episode}{6000}}. \tag{27}$$

This leads to ϵ starting at 1 (fully exploration) and ending at a 0.05 probability of a random action per time step. When the models are tested they will only choose greedy actions ($\epsilon = 0$).

All models include the auto-break shielding technique for faster and safer learning. They also use the kinematics observation space with 11 rows, i.e. observing in total 10 other cars and objects. The main hyperparameters used can be found in the appendix.

4 Experimental Results

In this chapter the results from the experiments will be presented in great detail. First the outcome of the training with different models will be shown. Thereafter, testing of the chosen model will be captured and explained with a few metrics of interest. In the next chapter the results will be discussed thoroughly.

4.1 Training Results

The models are all trained on 10 000 episodes with the ϵ-greedy method where ϵ follows equation 27. 10 000 episodes should be enough for the models to find a policy that can obtain high reward consistently. An episode is terminated if one car crashes or if the time steps exceed 100 (resulting in a maximum episode length of 50 seconds). At each episode a random number of RL driven cars are chosen between 5-10 and a random number of non-RL driven cars are chosen from the same interval. Hence, the maximum percentage of RL driven cars is 2/3 and the minimum is 1/3 during training.

Figure 4 shows the moving average, with a window of 1 000 episodes, of the models reward when training. All models learn at a fast pace until about episode 3 000, thereafter the learning plateaus. This indicates that the models will probably not be able to yield significantly higher rewards if they were to train more. The models reach an average reward level of around 25, this is considered to be decent, since the maximum reward is 100 and the minimum is -119. Meaning that the models reached 75% of the maximum reward, relative to the minimum. The model that performs best when training, i.e. reaching the highest moving average reward, is the EGO-DDQN and will thus be chosen as the model to test.

After the models have trained, the success rate for EGO-DDQN was captured. Remember that the success rate is the ratio of successful episodes, where an episode is successful if the cars do not crash. The EGO-DDQN reached a success rate of about 80% during 400 test-episodes.

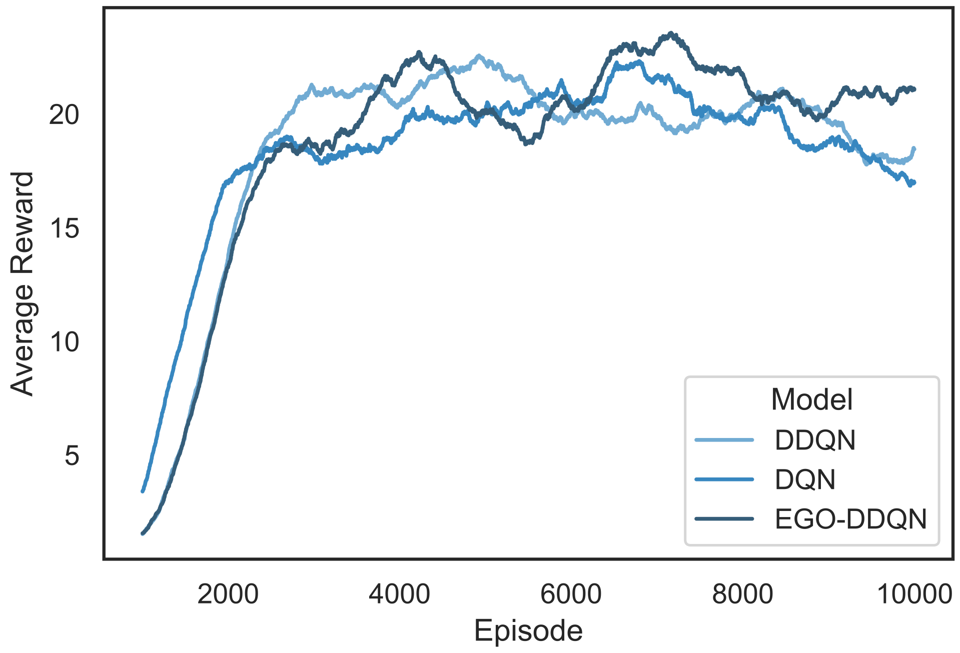

Figure 4: Moving Average of Reward with a Rolling Window of 1 000.

4.2 Test Results

Now that a robust and high performing (consistently high rewards) model has been successfully trained, the model is tested and evaluated in different settings. The testing will be performed on the same course as trained on, with a set number of 15 cars. First, all of the cars will be agents, i.e. have speed recommendations, and tested until 500 episodes are successful. An episode is not successful if one of the cars crashes. For each time step of each episode the congestion is measured and saved. Thereafter, the same procedure is done with 12/15 of the cars having the speed recommendations system. Then 10/15, 7/15, 5/15, 2/15 down to 0% of the cars having this system. This is done to capture the effect the speed recommendation system can have on different settings. To measure the congestion, the metric vehicle hours will be used. The vehicle hours will be measured by the average number of vehicle hours between meter 400 and 600. The distance 400 and 600 is determined as the distance of interest since the object that creates congestion is spawned at meter 500. The distance of interest is the part off the course where there the traffic is not in "free flow".

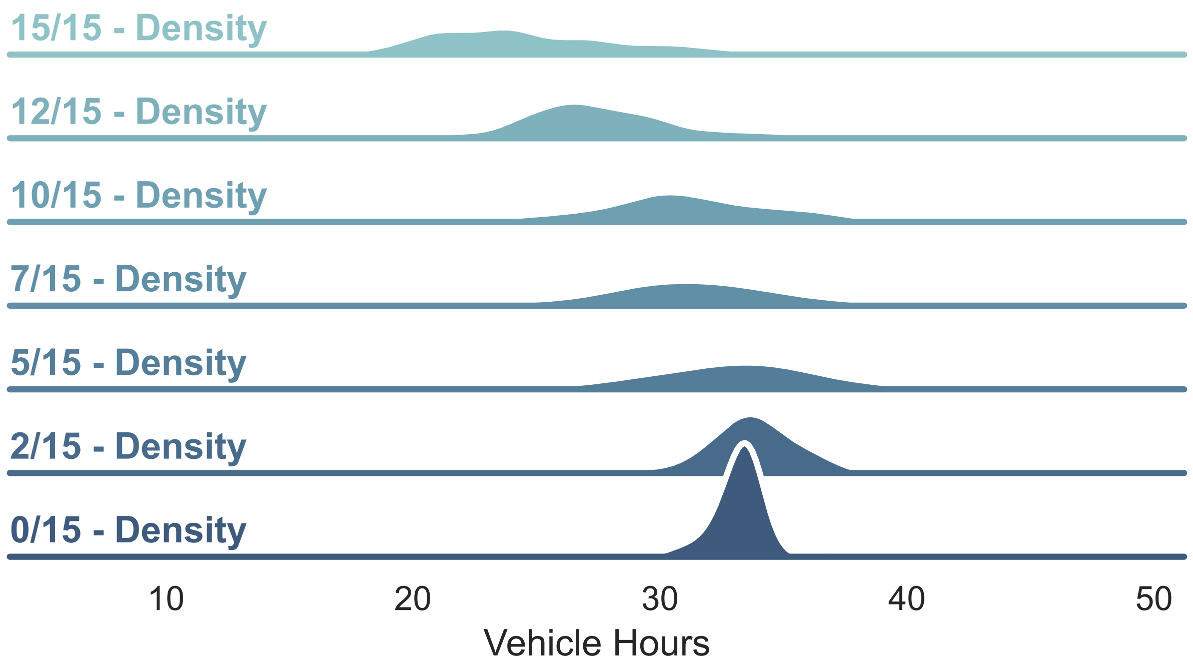

Figure 5: Vehicle Hours Densities for the Different Ratios of RL Driven Vehicles.

Figure 5 shows the vehicle hours densities for the different settings. Each density is named after the ratio it represent, for example, 12/15 represent that 12 out of the 15 cars had the RL speed recommendations system. From here on, these settings will be referred to as ratios. The figure shows a clear relationship with the number of cars being RL driven and the number of vehicle hours. The more RL driven cars, the less vehicle hours. One can also see that the variance tends to be bigger when the ratio of RL cars increases. This could be due to the added noise, that represent the game theory aspect of this thesis, from section 3.1. on the RL vehicles.

Since it is hard to capture the outliers in Figure 5, boxplots of the ratios are visualized in Figure 6. Here one can see that all ratios have a lot of outliers. This could indicate that the sample sizes are too small for the ratios to converge to their real distribution. However, a sample size of 500, where each episode is 100 time steps long, per ratio is considered big

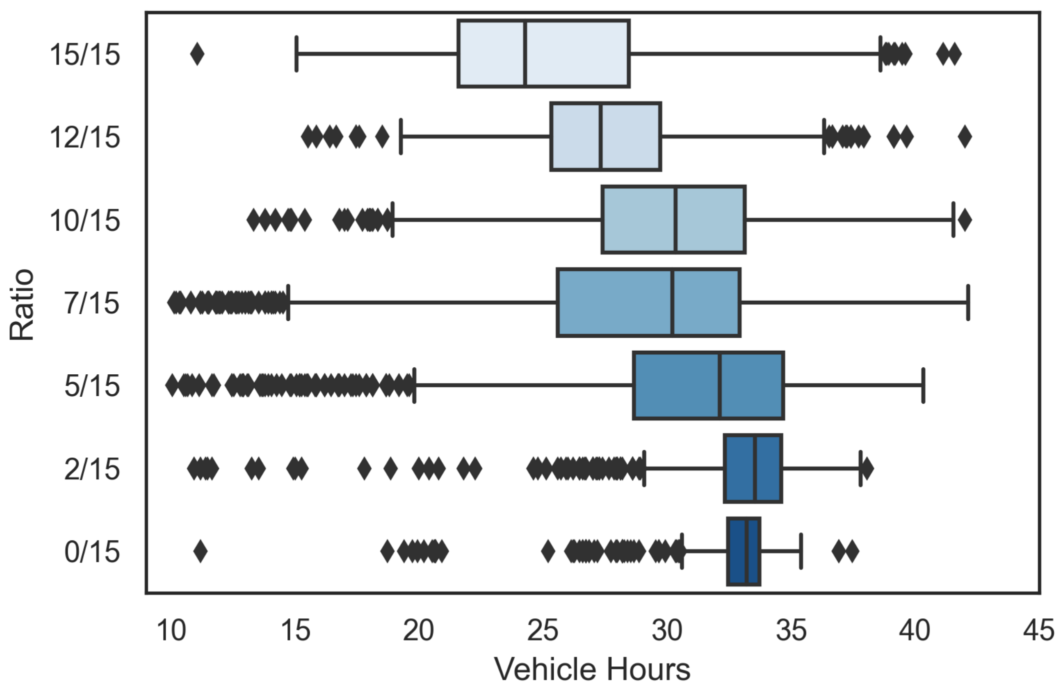

Figure 6: Boxplot of Vehicle Hours.

enough. Nevertheless, the outliers should be in mind when drawing conclusion about the results.

A two-sample Kolmogorov-Smirnov test on the vehicle hours metric is performed for each possible pair. This is done to see if any ratio is performing similar enough, to another ratio, for us to be able to conclude that they come from the same distribution. Would for example the setting with only RL driven cars and the setting with non RL driven car (baseline model) get an insignificant p-value on the two-sample Kolmogorov-Smirnov test, then it would imply that the chosen RL model is not better nor worse than the baseline, even if the means would imply otherwise.

Table 2: P-values for Two-sample Kolmogorov–Smirnov Tests on Vehicle Hours Samples.

Ratio	15/15	12/15	10/15	7/15	5/15	2/15	0/15
15/15	1						
12/15	0.00	1					
10/15	0.00	0.00	1				
7/15	0.00	**0.01**	0.00	1			
5/15	0.00	0.00	0.00	0.00	1		
2/15	0.00	0.00	0.00	0.00	0.00	1	
0/15	0.00	0.00	0.00	0.00	0.00	0.00	1

As one can see in Table 2, all of the p-values are considered significant (on a 5% significance level). This indicates that non of the vehicle hours ratios are alike. Nevertheless, to further investigate the vehicle hours, Table 3 shows some statistics of interests $\big($Mean, Standard Deviation (SD), Min, Max and 95% Confidence Interval (CI 95%)$\big)$.

Table 3: Vehicle hours - Statistics of Interest.

Ratio	Mean	SD	Min	Max	CI 95%
15/15	25.62	5.16	11.06	45.47	[25.16; 26.07]
12/15	27.68	3.95	15.53	42.00	[27.33; 28.03]
10/15	29.75	5.20	13.33	42.00	[29.30; 30.21]
7/15	28.21	7.14	10.13	43.60	[27.58; 28.84]
5/15	30.27	6.86	10.06	44.93	[29.66; 30.87]
2/15	32.72	3.97	10.93	38.06	[32.38; 33.07]
0/15	32.50	2.65	11.20	37.46	[32.27; 32.73]

Table 3 shows that the means indeed decrease when the ratio of RL driven cars increases as discussed earlier. Comparing the ratio with only RL driven cars with the baseline model, the RL speed recommendation can reduce the congestion with 21%. Already when there are 5 out of 15 cars that use the speed recommendation, the congestion will decrease with nearly 7%. The differences in the means will later in this section be statistically tested.

As for the standard deviations, when the ratio increases, the standard deviation increases. This seems to have an effect on the minimum and maximum for each ratio, since these values are the most extreme for the ratios with the highest standard deviation. The confidence intervals does mostly not intersect with each other. 0/15 with 2/15, 5/15 with 10/15 and 7/15 with 12/15 being the exceptions. This indicates that the means are not the same.

To test whether the means are alike or not, an ANOVA will be performed. Although, since the variances seem to violate the one-way ANOVA assumption of homoscedasticity a levene test (that tests if the variances of samples are equal) was performed. The test rejected the null, the the variances are equal, with a p-value > 0.001. Thus, Welch's ANOVA will be performed instead. The test assumes independent sampling and that the distribution of the residuals are normal. After every simulated episode, the environment resets, making the sampling independent. The sample size for each ratio is 500 and we can on that account rely on the central limit theorem when it comes to the normality assumption. Additionally, a visualisation of the residuals (see appendix) does not give us enough evidence to disregard that the distribution of the residuals is a normal distribution. The assumptions are therefore assumed to be fulfilled and a Welch's ANOVA test can be conducted.

Table 4: Welch's ANOVA results.

Groups	DF_1	DF_2	F-score	P-Value	N per group
Congestion Ratios	6	1530.63	200.245	1.5233e-188	500

Table 4 shows that the Welch's ANOVA test returned a significant p-value of 0.001, showing that at least one mean is statistically different from the other means. The Games-Howell post-hoc can consequently be performed to identify how the means differ from each other.

Table 5 shows the p-value for each pair of sample means. The only means that do not significantly differ from each other are the pairs 12/15 with 7/15, 10/15 with 5/15 and 2/15 with 0/15. This statistically ensures us that most means are not alike. Only one mean is similar to the baseline model (0/15) showing that the RL speed recommendations with a ratio above 2/15 have a statistical significant effect on the congestion mean. Also ensuring that when the ratio increases, the congestion mean decreases (ratio 2/25 being the exception).

Table 5: P-values for Games-Howell Post-hoc Test on Vehicle Hours Samples.

Ratio	15/15	12/15	10/15	7/15	5/15	2/15	0/15
15/15	1						
12/15	0.00	1					
10/15	0.00	0.00	1				
7/15	0.00	**0.74**	0.00	1			
5/15	0.00	0.00	**0.82**	0.00	1		
2/15	0.00	0.00	0.00	0.00	0.00	1	
0/15	0.00	0.00	0.00	0.00	0.00	**0.9**	1

5 Discussion

The training of the models showed that they all could learn fast and become habile performing models. They reached a relatively high average reward compared to the maximum and minimum potential. Nonetheless, the models would still need to improve a lot to reach the maximum reward. This is probably because of the complexity of the course and the design of the reward function. It could be that it is impossible to maximize the reward function. For example, it gives more reward when driving on the right side and objects could spawn on the right side forcing the vehicles to drive on the left. Making it in this example impossible to maximize the reward. With this in mind, the models are considered to be useful.

The testing of the EGO-DDQN showed two important things. Firstly, and most importantly, the congestion decreases when the ratio of RL cars increases. The baseline model without any RL cars had a mean vehicle hours of 32.5 on the course. All ratios except 2/15 had a smaller mean and they could be statistically ensured to be different from the baseline mean. This shows that the model can reduce the congestion. In our case, the model can reduce congestion up to 21%. If the model can reduce congestion in other settings and how much, we cannot answer since there needs to be more testing. However, this is a good indication that using RL in the traffic can enhance the traffic flow and driving experience.

Secondly, the variance of the congestion increased when the ratio of RL cars increased. This could be due to three reasons. Either the artificial noise on the RL cars cause the increase of variance. Since the cars did not follow the speed recommendations fully, this could lead to unnecessary queues and thus congestion. Although, when training, the models should learn to take this into account. Furthermore, Figure 4 shows that the models hit a plateau and could not learn anymore. This indicates that the second reason is more plausible, that is, that the model take high risks in order to maximize the reward. Taking a risk could lead to less congestion but also to more congestion. Nonetheless, keep in mind that the model perform better than the baseline. The third reason could be that the reward function is not optimized for the problem and that there are many ways to yield high reward. Some of those could heavily decrease congestion, some do not. However, inspecting the reward function (equation 25 and 26), it is hard to visualize a scenario where an increased congestion would yield high reward.

When scaling up the number of cars in the simulations (to more than 40), the ratio of successful episodes when testing the models decreased. This is possibly because of the complexity increasing exponentially and thus needing a more complex model and/or exponentially more training. Increasing the complexity of the course could create a similar problem. The more complex course the worse model performance. The problem with a ratio of low successful episodes was partially solved by including the shielding technique in the model. The shielding technique increased the success rate from .5 to .8. If a RL speed recommendation would be implemented in reality, a success rate of .8 would be too low and dangerous. The success rate should be 1 or extremely near 1 in all possible settings. Hence, this is the biggest **drawback** with RL speed recommendations. Two possible ways to solve this are; a more complicated RL model trained with immense computational power or implementing other non-RL safety systems, as shielding. This could lead to a robust system. Yet, it is important to note that the focus in this study is not the safety of RL speed recommendation. It is whether it can reduce congestion or not.

As already mentioned, the Gaussian noise seems to have an effect on the variance of the congestion. The effect it has on the mean of the congestion has not been captured. One could train a model without noise to compare it with the result of this thesis. Nonetheless, the results were statistically tested and showed that the congestion could be reduced with RL speed recommendations, even though a noise were added implying that the speed recommendations was not fully followed. This shows that a speed recommendations system could still work in real traffic, where the drivers do not fully obey the recommendations. Thus, the models can take into consideration some game theory aspects that occur in real traffic.

6 Conclusion

The aim of this thesis is to study if reinforcement learning speed recommendations could reduce traffic congestion. This has been done by simulating a highway with obstacles to artificially create congestion. Three different models were trained, a Deep Q-Network and two Double Deep Q-Networks, for 10 000 episodes. The Double Deep Q-Networks called EGO-DDQN performed the best with an average reward of 25 (75 percentile) and a success rate of .8, which we concluded is a useful model. After finding a model that yields consistently high rewards, the model was tested. The testing was performed by simulating successful episodes and capture the congestion. The congestion, measured in vehicle hours, was captured on seven different settings where the ratio of RL driven car was the difference between the settings. The testing showed that the RL model succeeded with the goal of reducing congestion, with the cost of higher variance. However, even with high variance, the model almost always outperforms the baseline model with no RL driven cars. This leads us to conclude that the aim of study has been successfully answered. MARL can be applied to speed recommendations on a vehicle-level to reduce congestion and control the traffic flow.

Nevertheless, even though a good model was found, there are some improvements that could have been done. First, the model has a unsatisfactory success rate, meaning that the ratio of episodes that prematurely terminate due to a crash, is too high. This could potentially be solved by more training or a non-RL safety mechanism. Second, the models' performance is highly dependent on the settings. When the number of cars are under 20 and the objects are few, the model performs well. Although, when increasing the number of cars and objects, the performance declines. This is probably due to the model not being robust enough. The solution is, as with the success rate, more training and/or a more complex model.

6.1 Future Work

There are endless possibilities to incorporate RL to speed recommendations. With the rise of 5G, the study of RL speed recommendations directly to vehicles should continue. This thesis focused on if RL speed recommendations on few cars could reduce the congestion. In the future, it would be interesting to test this idea on many cars, say over 200, on different courses or even in a traffic network. Also including fundamental diagram as a measure of congestion. This could enhance the robustness of the findings in this thesis.

Furthermore, to study speed recommendations on a continuous scale or at least rounded to the nearest integer would be an interesting aspect. In this thesis, the speed recommendations are in 5 km/h intervals. However, some situations call for a more specific recommendation and more precision could be beneficial to the congestion and traffic flow.

To Remove the actions that allow the RL agent to change lane and create a change lane mechanism that copies how humans change lanes, would be an interesting aspect. There, the RL only gives speed recommendations and does not focus on the change lane aspect.

Finally, the last idea is to study the possibility of a RL speed recommendation system that takes emergency vehicles into considerations. This could potentially lead to less congestion for the emergency vehicles when duty calls, with the cost of more congestion for the rest of the traffic.

Acknowledgements

Foremost, I would like to offer my special thanks to Jieqiang Wei and Ronnie Pingel for their assistance during the research project. Besides my supervisors, I would like to thank the stakeholders at Ericsson for providing the research question and helpful insight regarding the subject.

References

[1] "Stockholm traffic report: Tomtom traffic index." [Online]. Available: https://www.tomtom.com/en_gb/traffic-index/stockholm-traffic

[2] M. Barth and K. Boriboonsomsin, "Real-world carbon dioxide impacts of traffic congestion," *Transportation Research Record*, vol. 2058, no. 1, pp. 163–171, 2008. [Online]. Available: https://doi.org/10.3141/2058-20

[3] B. Katz, J. Ma, H. Rigdon, K. Sykes, Z. Huang, K. Raboy, and J. Chu, *Synthesis of variable speed limit signs.* United States Department of Transportation, Federal Highway Administration, 2017.

[4] M. Wiering, "Multi-agent reinforcement learning for traffic light control." 01 2000, pp. 1151–1158.

[5] E. Walraven, M. T. Spaan, and B. Bakker, "Traffic flow optimization: A reinforcement learning approach," *Engineering Applications of Artificial Intelligence*, vol. 52, pp. 203–212, 2016.

[6] Z. Li, P. Liu, C. Xu, H. Duan, and W. Wang, "Reinforcement learning-based variable speed limit control strategy to reduce traffic congestion at freeway recurrent bottlenecks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 11, pp. 3204–3217, 2017.

[7] C. Wang, J. Zhang, L. Xu, L. Li, and B. Ran, "A new solution for freeway congestion: Cooperative speed limit control using distributed reinforcement learning," *IEEE Access*, vol. 7, pp. 41 947–41 957, 2019.

[8] Y. J. Ha, S. Chen, J. Dong, R. Du, Y. Li, and S. Labi, "Leveraging the capabilities of connected and autonomous vehicles and multi-agent reinforcement learning to mitigate highway bottleneck congestion," 10 2020.

[9] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction.* The MIT Press., 2018.

[10] R. Bellman, “A markovian decision process,” *Journal of mathematics and mechanics*, vol. 6, no. 5, pp. 679–684, 1957.

[11] S. B. Thrun, “Efficient exploration in reinforcement learning,” 1992.

[12] L. Buşoniu, R. Babuška, and B. De Schutter, “Multi-agent reinforcement learning: An overview,” *Innovations in multi-agent systems and applications-1*, pp. 183–221, 2010.

[13] Y. Yang and J. Wang, “An overview of multi-agent reinforcement learning from game theoretical perspective,” *arXiv preprint arXiv:2011.00583*, 2020.

[14] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, “Safe reinforcement learning via shielding,” vol. 32, Apr. 2018.

[15] C. J. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[16] H. Hasselt, “Double q-learning,” *Advances in neural information processing systems*, vol. 23, pp. 2613–2621, 2010.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, “Human-level control through deep reinforcement learning,” *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[18] H. van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double q-learning,” 2015.

[19] J. Zhang, H. Chang, and P. A. Ioannou, “A simple roadway control system for freeway traffic,” in *2006 American Control Conference*. IEEE, 2006, pp. 6–pp.

[20] C. Feliciani, H. Murakami, and K. Nishinari, “A universal function for capacity of bidirectional pedestrian streams: Filling the gaps in the literature,” *PLOS ONE*, vol. 13, p. e0208496, 12 2018.

[21] F. J. Massey Jr, “The kolmogorov-smirnov test for goodness of fit,” *Journal of the American statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.

[22] R. G. Miller Jr, *Beyond ANOVA: basics of applied statistics.* CRC press, 1997.

[23] B. L. Welch, "On the comparison of several mean values: A alternative approach," *Biometrika*, vol. 38, no. 3-4, pp. 330–336, 12 1951.

[24] R. E. Kirk, "Experimental design," *Handbook of Psychology, Second Edition*, vol. 2, 2012.

[25] E. Leurent, "An environment for autonomous driving decision-making," https://github.com/eleurent/highway-env, 2018.

[26] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, pp. 1805–1824, 02 2000.

[27] OECD, "Report on speed management measures," *Organisation for Economic Co-operation and Development, Paris.*

[28] E. Leurent, "rl-agents: Implementations of reinforcement learning algorithms," https://github.com/eleurent/rl-agents, 2018.

[29] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Machine learning*, vol. 37, no. 3, pp. 277–296, 1999.

[30] E. Leurent and J. Mercat, "Social attention for autonomous decision-making in dense traffic," *arXiv preprint arXiv:1911.12250*, 2019.

Appendix

Proofs & Settings

The full proof of how a Q-learning model overestimates the action values, mentioned in section 2.1.6, will in this section be presented. Remember that the Q-learning algorithm, equation 5, is defined as

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_{t+1}, A_t) \Big].$$

Q-values are noisy. Algorithm takes the max over all Q-values, thus it is probable to get an overestimated value. For example, the expected value of a dice roll is 3.5. However, if one would roll the dice 100 times, then the max over all throws is probably greater than the expected. If all action values were equally overestimated, this would not be a problem. This is not the case. The overestimations are not uniform, which slows down learning since the time spent exploring will be longer than necessary.

In section 3.1, the shielding technique used in the simulations are described. One component in the shielding technique is the recommended acceleration calculated by the IDM model [26]. To further explain the recommended acceleration, the acceleration is calculated by

$$v = \frac{dv}{dt} = \alpha \left(1 - \left(\frac{v}{v_0} \right)^{\delta} - \left(\frac{s^*(v, \Delta v)}{s} \right)^2 \right),$$

with

$$\frac{s^*(v, \Delta v)}{s} = s_0 + vT + \frac{v \times \Delta v}{2\sqrt{ab}},$$

where s_0 a set minimum desired net distance (A vehicle can't move when the distance from the vehicle in front is less than s_0), T the set minimum possible time to the vehicle in front, α the maximum acceleration, b is the comfortable braking deceleration. Thus, v is the recommended acceleration used in the shielding technique used.

In section 3.2. the reward function used in the thesis is explained. Remember that the scaled variables used in the function are $\mathsf{s}_s, \mathsf{n}_s, \mathsf{c}_s$. And the boundaries for the variables are

- s: $\mathsf{s}^q_{min} = 25$ and $\mathsf{s}^q_{max} = 35$, $\mathsf{s}^s_{min} = 0$ and $\mathsf{s}^s_{max} = 10$
- n: $\mathsf{n}_{min} = 5$ and $\mathsf{n}_{max} = 15$
- n: $\mathsf{c}_{min} = 0$ and $\mathsf{c}_{max} = 1$

The models used in this thesis are built on different layers. These layers are the perceptron layer and the attention layer. The perceptron is an algorithm for binary classifiers, where the classifier determines whether or not a vector of numbers, or number, belongs to a specific class. It is a type of linear classifier that performs prediction through a linear function combining weights with the features [29]. The attention layer helps the model to filter data and keeps only what is relevant for an action. Thus, the agent should focus on close vehicles or objects in the way of the planned route [30].

The main hyperparameters in the models are

- Exploration method. For example, ϵ-greedy.
- Observation space. For example, kinematics.
- ϵ: Small number, deciding the frequency of random actions during training.
- Maximum number of cars to observe in the observation space.
- Learning rate: Controls how much to change the model in response to the estimated error.
- Number of episode to train.

Plots

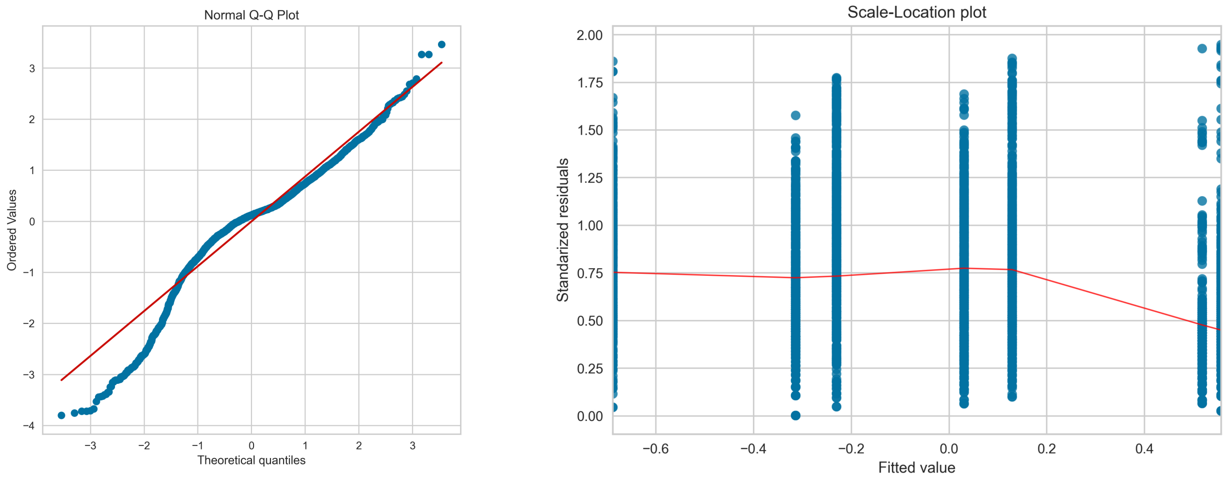

Figure 7: Plots of the residuals.