Evaluating the Robustness of DQN and QR-DQN in Traffic Simulation

Analyzing the Effect of Quantile Manipulation in Environmental Variability

Cristian Toadere[1]

Supervisor(s): Dr. Frans Oliehoek[1]**, Dr. Mustafa Celikok**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2025

Name of the student: Cristian Toadere
Final project course: CSE3000 Research Project
Thesis committee: Dr. Frans Oliehoek, Dr. Mustafa Celikok, Dr. Annibale Panichella

Abstract

As autonomous driving systems advance, ensuring the robustness of underlying decision-making algorithms becomes increasingly critical. This study assesses the performance and reliability of two reinforcement learning models, Deep Q-Network (DQN) and Quantile Regression DQN (QR-DQN), within the context of a simulated highway environment. While DQN has been widely adopted for its simplicity and effectiveness in discrete action spaces, it suffers from overestimation bias and lack of performance in out-of-distribution environments. QR-DQN addresses some of these limitations by modeling the distribution over returns using quantile regression, offering a superior representation of uncertainty. This research focuses on two core objectives: (1) implementing a risk-averse decision-making strategy using the quantiles of QR-DQN to enhance safety and reliability, and (2) evaluating the robustness of DQN and QR-DQN as the test environment deviates from training conditions. Results show the limitations of DQN and demonstrate QR-DQN's higher robustness in different environments. Moreover, a better performing alternative of QR-DQN is presented, employing a conservative behaviour through the use of its quantiles. This puts emphasis on the implemented model's trade-off between maximising rewards and avoiding collisions, providing a safer approach.

1 Introduction

With the field of Autonomous Driving (AD) advancing in recent years [1], the need for reliable and robust Machine Learning algorithms becomes more significant. Deep Reinforcement Learning (Deep RL) has been showing considerable promise in the development of intelligent driving agents that are capable of navigating complex traffic scenarios [2]. A widely adopted Deep RL approach is Deep Q-Network (DQN) [3], capable of effective operation in environments with discrete action spaces [4]. However, it suffers from overestimation bias, a phenomenon in which the agent systematically learns to overestimate the true values of certain actions, caused by the maximisation operator. Moreover, performance reduction can be seen in out-of-distribution or adversarial scenarios [5].

In order to address these limitations, one distributional model introduced is Quantile Regression Deep Q-Network (QR-DQN) [6]. Unlike DQN, which estimates a single expected return (i.e. the mean of the return distribution), this model uses quantile regression to estimate the quantile values of the return distribution. Although this allows for a more comprehensive understanding of the return distribution, QR-DQN still computes the mean when choosing its actions. Without utilising the quantiles when deploying the policy, the model becomes similar to DQN. However, the quantiles can be used to, in fact, train policies with different risk sensitivities. Even though QR-DQN has shown superior results in benchmark tasks compared to DQN, its effectiveness under variable and progressively differing environmental conditions has not been sufficiently explored. Specifically, there is room to understand how the quantiles of QR-DQN can be utilised to improve safety and robustness through employing a risk averse approach.

This study aims to evaluate the robustness of these models in a highway scenario using HighwayEnv, an AD simulation environment [7]. This environment supports Deep RL model training and testing, allowing environment parameter configuration, which makes it suitable for this experiment. The focus of this research is two-fold: (1) analysing the results of implementing a conservative decision-making approach utilising QR-DQN's quantile distribution, and (2) measuring performance degradation of DQN, QR-DQN and the proposed implementation in test environments which differ from the training setting. Through this, the goal is to contribute to the understanding of each model's performance and robustness in dynamic environments and provide an implementation that produces better performance by using QR-DQN's quantiles.

In order to conduct this study, a background research on the models used and related works is done to understand the current state of this topic. After that, an experiment is carried out by training and testing the models in the highway environment. Firstly, the Risk-Averse QR-DQN is implemented, presenting the modification made from the original model. Then, the environment configuration is adapted for this experiment, defining the parameters that will later be changed in testing. Lastly, the models are trained using multiple seeds, each of them then tested in various environments to assess their robustness.

The subsequent sections are organized as follows. Section 2 offers an overview of the models used in this study and summarises some related works. Section 3 describes the methodology used for this experiment, explaining the configurations used for the models and environments. It further provides insights into the custom QR-DQN policy implementation. Following that, Section 4 presents the results from training and testing and discusses the findings. Finally, Section 5 offers the conclusion and a look into possible future work.

2 Literature Review

In this section, an overview of the models used in this experiment are presented. In addition, related experiments or studies are mentioned, giving a summary of their results and findings.

2.1 Deep Q-Networks

The Deep Q-Network (DQN) is fundamentally based on the Q-Learning algorithm [8]. This algorithm aims to iteratively learn an optimal action-value function $\boldsymbol{Q(s, a)}$, that maximises the expected future reward for taking action $\boldsymbol{a}$ in given state $\boldsymbol{s}$. Its return is an estimate of the expected cumulative reward for taking action a in state s. However,

due to its requirement of maintaining a value table for all state-action pairs, Q-Learning becomes inefficient in dealing with large or continuous state spaces.

To overcome this limitation, Mnih et al. [3] and [9] introduced the DQN model, which makes use of deep neural networks to approximate the Q-function. Its introduction represented a major breakthrough in Deep RL, enabling agents to effectively learn policies straight from high-dimensional input. The underlying algorithm takes as input a state, returning a set of Q-values for all possible actions. Crucial to its stability and performance, DQN incorporates experience replay and target networks. Experience replay uses a memory buffer to store experiences at each step in the form of state, action, reward and next state. The training is done on random small batches of experiences from the buffer in order to break the correlation between consecutive data and improve sample efficiency. Target networks are introduced in DQN as secondary neural networks, used to calculate the target Q-values, being periodically updated to mitigate oscillations and divergence in learning. The Bellman's equation for DQN is showcased in equation (1),

$$Q(s,a;\theta) = r + \gamma \max_{a'} Q(s',a';\theta') \tag{1}$$

where:

- $Q(s,a,\theta)$ is the current state value with weight θ,
- *r* is the reward received for taking action *a* in state *s*,
- γ is the discount factor that determines how much future rewards are taken into account, and
- $\max_{a'} Q(s',a';\theta')$ is the maximum value for state *s'* considering all possible actions *a'*, having weight θ'.

To train the neural network, a loss function is defined by the squared difference of the Bellman equation, as in the following equation (2):

$$L(\theta) = \mathbb{E}\left[(r + \gamma \max_{a'} Q(s',a';\theta') - Q(s,a;\theta))^2\right] \tag{2}$$

Despite its performance and success, because of its reliance on the expected return, DQN has shown limitations in the form of overestimation bias and poor performances in uncertain environments [5].

2.2 Quantile Regression Deep Q-Network

To overcome the limitations of DQN in modelling uncertainty, Dabney et al. [6] proposed a distributional reinforcement learning approach, named QR-DQN. This new model approximates the value distribution over possible returns incorporating quantile regression, compared to the single expected return for each state-action pair provided by DQN. In their paper, Dabney et al. propose the following changes from the base model of DQN when implementing the new model. A distributional variant of the Bellman's optimality operator is used to estimate a state-action value distribution, as given in the following equation (3):

$$\mathcal{T}Z(x,a) = R(x,a) + \gamma Z(x',a'), \tag{3}$$
$$x' \sim P(\cdot|x,a), a' = \arg\max_{a'} \mathbb{E}_{z\sim Z(x',a')}[z].$$

Moreover, the DQN loss function is replaced by the Huber quantile loss (4), a variation of the Huber loss [10] defined as in equation (5).

$$\rho_\tau^\kappa(u) = |\tau - \delta_{\{u<0\}}|\mathcal{L}_\kappa(u). \tag{4}$$

$$\mathcal{L}_\kappa(u) = \begin{cases} \frac{1}{2}u^2, & \text{if } |u| \le \kappa \\ \kappa(|u| - \frac{1}{2}\kappa), & \text{otherwise} \end{cases}. \tag{5}$$

Finally, RMSProp [11] has been swapped with Adam [12] as the optimisation strategy. The output layer of the neural network architecture of DQN was also adapted to be of size $|\mathcal{A}| \times N$, where $\mathcal{A}$ is the action space and *N* is the quantile target hyperparameter. This approach has been shown to outperform DQN on the Atari 2600 benchmark [6] by learning a distribution over possible future rewards using multiple quantile regressions. Instead of considering only the expected return like DQN, it estimates the full possible return distribution by predicting quantile values. The network is trained to minimize the quantile regression loss, which compares the predicted quantiles to target quantiles using the Huber quantile loss. Through the use of quantiles, QR-DQN gathers more subtle information about the variability and risk of each action, making it more efficient in uncertain environments.

2.3 Related Studies

Prior to conducting the experiment, a review of existent studies and results was carried out. Most relevant studies found are summarised in this subsection. In one study [13], an experiment on DQN is performed using three environment based driving modes: safe, normal and aggressive, determined by the ego vehicle speed range. Each variant was trained for 12000 timesteps in HighwayEnv, then tested for 20 episodes. Results were measured by the percentage of completed episodes and total reward obtained. The safe mode achieved the highest reward (32) in training, followed by normal (28) and aggressive (26). This trend was confirmed in testing, where safe mode obtained highest reward (35.7), then normal (33.42) and lastly aggressive (31.85). However, completion rates were highest for aggressive mode (95.875%), then normal (94.625%) and safe (90.750%). The study shows that DQN struggles in challenging and dynamic driving scenarios.

Another study [14], examined DQN applied to autonomous vehicles in a highway simulation environment with varying levels of traffic density. Following training with 20000 timesteps and 40 vehicles, the model was then tested using in turns 20, 40, 60, 80 vehicles, for 20 episodes each. The results show a decline in total rewards with higher densities, with 20 vehicles/frame achieving 35.6117 and success rate of 90.075%, all the way down to 19.3024 and 61% for 80 vehicles. The study emphasises the limitations of DQN when it comes to adapting to denser environments.

In a related study [15], DQN and Proximal Policy Optimization (PPO) model are compared in increasing lane and traffic density scenarios in HighwayEnv. The experiment is undertaken using 2 to 8 lanes with vehicle count is 10 times

the lane count for each test. The results show DQN having an inconsistent performance as the number of lane increases, showcasing its limitations to perform in changed environments. However, PPO presents progressive improvements proving its efficiency and capability to navigate scenarios with higher number of lanes. These findings are confirmed in a different study [16], where the effectiveness of DQN and PPO was compared in different scenarios using the CARLA environment [17]. One of the scenarios was a highway with medium traffic density in foggy weather condition. The training and testing is done on a 70%/30% data split with no cross-validation. PPO outperformed DQN in all metrics, while requiring more computational load. The study highlights that PPO can get higher results, performing better than DQN in different scenarios.

One paper proposes the Implicit Quantile Network (IQN) [18] as a model that enables risk-sensitive policy learning. Unlike QR-DQN, which relies on fixed discrete approximations, IQN learns a continuous quantile function. This method improves fidelity and data efficiency, outperforming existing approaches in benchmark tasks. Despite its simple architecture, IQN allows for exploration of risk-sensitive behaviours, showcasing competitive results compared to state-of-the-art agents.

Most of the studies found and presented compare DQN against other models in different AD environments. However, a study that implements a risk-averse QR-DQN algorithm utilising the model's quantiles has not been found, a gap which this paper aims to cover. Furthermore, the goal is to compare this implementation's performance against the DQN and QR-DQN models.

3 Methodology

This section outlines the methodology taken in this study. It includes the environment configuration and experiment setup for comparing the robustness of DQN and QR-DQN in a highway simulation environment. Furthermore, the implementation of a risk-averse policy in the QR-DQN model is given, describing the changes made. The code used for running this experiment and all the necessary hyperparameters are accessible online, in a GitHub repository [19], which can be used for reproducing this study.

3.1 Risk-Averse QR-DQN

To promote a risk-averse behaviour in QR-DQN, the standard method of computing the expected return is modified. Where base QR-DQN averages over all quantiles of the predicted return distribution, the proposed implementation alters the algorithm to consider only a smaller, lower subset of the quantiles. This adjustment determines the policy to favour more conservative decisions, choosing actions with higher return in the tails of their return distribution. Thus, the agent aligns with a risk-averse approach while still utilising the distributional nature of QR-DQN.

The base models used in this study are taken from Stable Baselines 3 (SB3) [20], a Python library that offers reliable implementations of RL algorithms. In order to implement the risk-averse behaviour, a couple of changes need to be made to the QR-DQN model. These are made by extending the already existing classes and methods offered in the SB3 library. The base model takes the `'n_quantiles'` parameter which represents the number of quantiles QR-DQN tries to learn. To be able to control how much of the quantile is to be considered, the parameter `'quantile_fraction'` is introduced in the initialisation of the model. After verifying the value of the parameter (i.e., resides in the range of [0, 1]), the `'predict'` method takes it into account when producing its return value, ensuring that the model considers at least one quantile. The SB3 QR-DQN implementation uses the following equation (6) to predict next actions:

$$argmax_a\ (mean(n\_quantiles)) \tag{6}$$

which returns action a with the maximum q_value, that is calculated based on the quantile mean. In the proposed implementation, the return of the `'predict'` method is changed to incorporate the newly introduced parameter when determining the next action. as shown in equation (7).

$$k = n\_quantiles * quantile\_fraction,$$
$$argmax_a\ (mean(k)) \tag{7}$$

This change causes the quantile mean to be calculated over a reduced subset of quantiles, resulting in action a being determined by considering the maximum q_value over this lower range mean.

The simple nature of this implementation allows for easy manipulation of the quantile fraction that is considered when predicting next actions. This way, the `'quantile_fraction'` can be tweaked to find a balance between conservative behaviour and reward maximisation. The goal is to increase safety by lowering the collision rate with minimum effect on the performance of the model. To test this out, two models are used in this study, specifically with `'quantile_fraction'` values 0.1 and 0.4. This choice is made in order to compare two levels of conservativeness, determined by the scope and time constraint of this study.

3.2 Environment Configuration

Among the widely used environments for autonomous driving simulations, HighwayEnv [7] is chosen for this experiment. While it provides multiple scenarios, the focus of this study is on the highway scenario (Figure 1), which can be configured to fit different requirements.

Figure 1: Screenshot of HighwayEnv's default highway scenario

Using the base environment provided in the repository, named *'highway-fast-v0'*, further changes were made to fit our experiment description. The following parameters have been

configured from the default settings, used throughout the training phase:

- 'simulation_frequency'= 5 (Hz): sets the number of updates per second made during the simulation, allowing for less computation and a faster run;
- 'lanes_count'= 3: determines the number of lanes used in the simulation;
- 'vehicles_count'= 50: the number of uncontrolled vehicles rendered in a simulation;
- 'speed_range'= [14, 30] (m/s): the speed range the ego vehicle can drive in, set to represent common real-world highway speeds (50 to 108 km/h).

To study the robustness and performance of the models in progressively changing environments, we made independent changes in the environment variables, such as 'lanes_count', 'vehicles_density' and additionally, the 'other_vehicles_type' parameter. More details on the testing settings is provided in section 3.3, under Model Evaluation. The reward system was kept the same as in training to ensure a correct understanding of the model's robustness in new environments.

The possible actions the agent can take to explore and maximise rewards are: changing one lane to the left, changing on lane to the right, remain idle, accelerate or slow down. Reward values are kept the same as defined in the default configuration, with the total reward showcased in equation (8):

$$R_i = r_i^{right\_lane} + r_i^{collision} + r_i^{speed},$$
$$\text{Total reward} = \sum_{i=1}^{n} R_i \tag{8}$$

where:

- *r_rightlane* $\in \{0, 0.1\}$: reward given to the agent when the vehicle is on the rightmost lane;
- *r_speed* $\in [0, 0.4]$: reward obtained based on speed, mapped to the ego vehicle's speed range configured previously;
- *r_collision* $\in \{0, -1\}$: reward given if ego vehicle crashes into another vehicle;
- *i* = current timestep.

Lastly, the elements of randomness are explicitly controlled and documented, making it easier for the study to be reproduced. To keep it simple and consistent, during each run the same seed as the model's is used for setting seed values across relevant libraries and components. This includes the random seeds for the Python module, NumPy, TensorFlow and PyTorch, alongside CUDA-level seeds that are initialised via their provided functions. In the case of PyTorch, deterministic behaviour is enforced while disabling benchmarking, which prevents PyTorch from trying different convolution algorithms and picking the fastest one for the hardware used. Furthermore, the 'PYTHONHASHSEED' OS variable along with the HighwayEnv's environment, action space and observation space are all configured to use the same seed. This comprehensive procedure helps in eliminating variability in each run that could arise from uncontrolled sources of randomness.

3.3 Experiment Layout

Training Setup

In order to ensure coherent results, each model is trained across 5 independent runs, each with the following different, randomly selected seeds: 13, 5728, 896, 3988, 73310. This approach is common in reinforcement learning, employed to guarantee reproducibility of results. By averaging results over multiple seeds, a more trustworthy indicator of how the model is expected to perform is provided. To help with consistency, the model seed used for each run is also used for the environment configuration as explained in the previous section.

The DQN and QR-DQN implementations used are taken from Stable Baselines 3 (SB3) [2], a Python library that offers reliable implementations of RL algorithms. Having an extensive documentation and a consistent interface, it provides effortless integration in the experiment codebase. Both models are trained using the same hyperparameters, the main difference being that QR-DQN takes the number of quantiles as a parameter. The Risk Averse QR-DQN model takes the quantile fraction variable in addition when initialised. For this experiment, the hyperparameter configuration used is presented in Table 1. Every model and seed combination is then trained using 50000 timesteps in order to ensure sufficient learning, while avoiding overfitting.

Model Evaluation

The testing setup is done in the same fashion as the training one, evaluating all models on same seeds (i.e. 13, 5728, 896, 3988, 73310), making it more convenient and simple to reproduce. Testing over multiple seeds ensures consistent results, avoiding biases and providing better benchmarking of each model's performance.

The evaluation is done in diverse environments to understand the capabilities of each model and how they compare in performance. The first test is done using the training environment, determined by the off-policy nature of the algorithms. Meaning that, the models use a different policy in training (i.e. random exploration) compared to the one used in testing (i.e. taking action with highest predicted value). This reinforces the need to assess the model's convergence and whether the agent has learned an effective policy. On top of the training environment, further four environments were designed, by manipulating one of the environment parameters listed below. Each testing environment is defined by the following changes:

- First environment sets variable 'lanes_count' to 4;
- Second environment sets variable 'lanes_count' to 6;
- Third testing environment changes the surrounding vehicles' behaviour to an aggressive one by setting the 'other_vehicles_type' parameter to "highway_env.vehicle.behavior.AggressiveVehicle", as defined in the HighwayEnv package;

Table 1: Hyperparameters and their descriptions for the SB3 models used

Hyperparameter	Value	Description
policy	MlpPolicy	Type of policy the model employs: multilayer perceptron in this case
net_arch	[256, 256]	Define architecture of the policy network: two hidden layers with 256 nodes each
learning_rate	5e-4	Step size for updating the network weights
buffer_size	50000	Size of the experience replay buffer
learning_starts	200	Step from which the model starts learning
batch_size	32	Number of samples drawn from the buffer during each update
gamma	0.8	Discount factor for future rewards: less than 1 prioritizes short-term rewards
train_freq	1	Environment step frequency the model should be trained at
gradient_steps	1	Gradient update rate per training step
target_update_interval	50	Frequency to update the target network
verbose	1	Logging to track training progress
seed	seed number	Sets the random seed used within the model
n_quantiles*	50	Number of quantiles used to approximate return distribution
quantile_fraction**	0.1 and 0.4	Fraction of each quantile to be considered when predicting next action

* Parameter used only for the QR-DQN models
** Parameter used only for the implemented Risk-Averse QR-DQN model

- Fourth test environment is defined by increasing the `'vehicles_density'` parameter to 1.5 from the default value of 1.0, to simulate a higher traffic scenario.

For every environment setting, each model-seed configuration is tested on 1000 episodes, ensuring a comprehensive pool of results. Each episode is configured to take the episode's index as its seed, making it easier to set up and reproduce the test. Upon completing a run, the mean and standard deviation of both the reward and episode length are saved, alongside the number of collisions, which are the evaluation metrics for evaluating and comparing the models. Collisions are determined by looking at the episode's info variable for the 'crashed' value. Episode reward is calculated in the same way as described in subsection 3.2.

4 Results

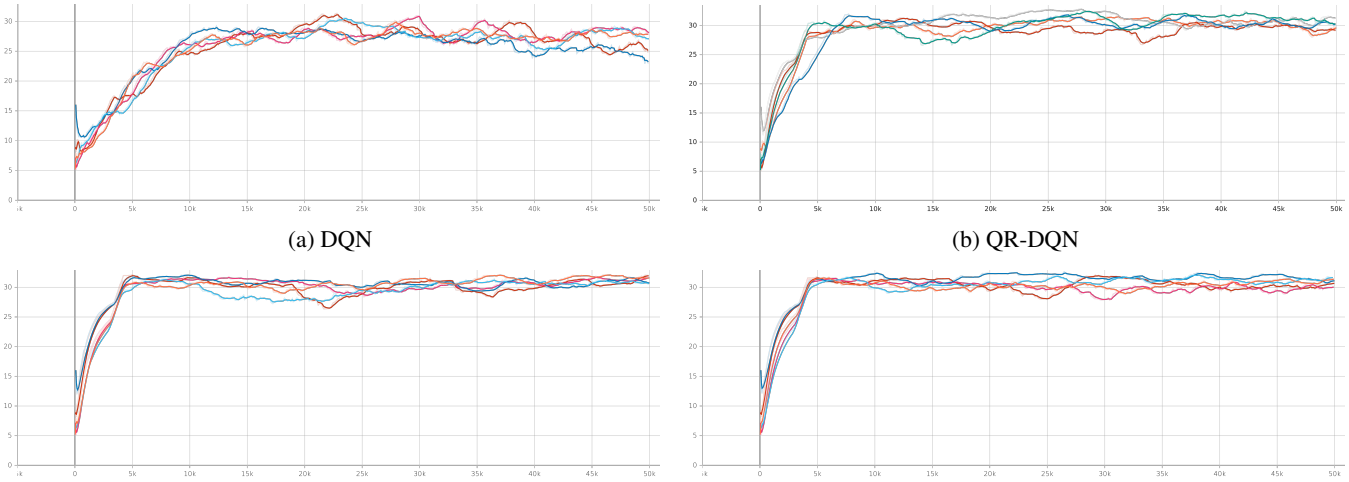
In this section the results of both the training and testing phases are presented, discussing the outcome and what it means. The experiment was carried out on the researcher's local machine, using a NVIDIA GeForce RTX 3070 Ti Laptop GPU for running the tasks. More result data can be found in the aforementioned Github repository [19].

For the remainder of this section, besides the standard naming convention used for the base models, the variations introduced in this experiment will be referred to as `RA QR-DQN 0.1` and `RA QR-DQN 0.4`, respectively, for the Risk-Averse QR-DQN implementation with the `'quantile_fraction'` parameter value set to 0.1 and 0.4.

4.1 Training Phase

Each model is trained for 50000 timesteps using the 3 lane highway scenario, as presented in section 3.2. In order to visualise the results, the Tensorboard logging feature is used, which tracks both the reward value and the episode length value each step. The reward value graphs for each model can be seen in Figure 2, each graph containing the results from 5 seeds of the indicated model. There is an intuitive correspondence between the progression trend of the reward value and length value (i.e., the longer the ego vehicle survives, the higher the reward it gets), hence only one metric is presented.

During training, DQN (Figure 2a) sees a stable progression until reaching timestep *10K*, after which the performance stabilises between reward values *25* and *30*, with a few seeds fluctuating. Towards the end of training there is a dip in results from a couple of seeds, decreasing the overall performance. The QR-DQN models (Figures 2b, 2c and 2d) showcase a faster and more consistent learning progression, all reaching a steady performance around *5K* timesteps. From there on, the reward value achieved fluctuates around the *30* mark, showcasing the exploration nature of the model. DQN achieves a mean reward value of *26.065 ($\pm$ 0.959 standard error)*, having a lower performance than the standard QR-DQN (Figure 2b), which reaches the mean reward value of *30.109 ($\pm$ 0.348)*. This indicates not only the limitations of DQN in training, but also the advantage QR-DQN has by capturing the quantile-based approximation of the return distribution.



(a) DQN (b) QR-DQN

(c) RA QR-DQN 0.1 (d) RA QR-DQN 0.4

Figure 2: Training reward value graphs for each model, shown as reward value per timestep.

Both variants of Risk-Averse QR-DQN outperformed the standard QR-DQN baseline, with *31.320 (± 0.304)* for RA-QRDQN 0.1 (Figure 2c) and *31.031 (± 0.267)* for RA-QRDQN 0.4 (Figure 2d). The improved performance justifies the benefit of incorporating risk-sensitivity by considering only a fraction of the model's quantiles. The 0.1 variant, which places a greater emphasis on lower quantiles, yielded the best performance, suggesting that a more conservative policy during training is favourable in this environment.

4.2 Testing Phase

Following training, the models were tested for 1000 episodes in 5 environments, as described in Subsection 3.3. In this subsection, the test environments defined are denoted as: `'3 lanes'` is the training environment, `'4 lanes'` and `'6 lanes'` are the first and second test environment, `'aggressive'` is the third one, which changes other vehicles' behaviour and lastly, `'traffic'` for the fourth testing environments, that increases the vehicle density. The evaluation metrics are determined by the average reward achieved over all the episodes and the number of episodes that ended in a collision. The results overview can be seen in Table 2 and Table 3, representing the mean and standard error of both metrics over 5 seeds for each model.

Results from the `3 lanes` environment demonstrate the off-policy character of the models, performing better in testing, by taking the action with the highest predicted value compared to the random exploration done in training. In the case of DQN, a slight decrease in performance can be seen in Table 2 from an environment to the other, putting further emphasis on the overestimation bias the model suffers from. These results correlate directly with the collision rate from Table 3, which see an increase over each environment. As expected, QR-DQN shows better performance for both metrics in comparison to DQN in the first 3 out of 5 environments, especially when it comes to collision rates. This reinforces the results found in training, where the quantile representation helps the model achieve better results by having a better understanding of the return distribution, which further improves collision avoidance. The reduced performance in the `aggressive` and `traffic` environments, when compared to DQN, can be explained by insufficient learning, although this is not proven in this study.

Among the QR-DQN models, the risk-averse variants show an increase in performance over the standard model across both metrics, while proving to be more predictable as showcased by the standard deviation in results. The performance can be explained by the conservative approach employed by the models through the use of the lower part of the quantiles when taking an action. Variation in results between the two RA QR-DQN models highlights a trade-off between enabling broader action exploration, and narrowing distributional focus via the lower quantile fraction. Both RA QR-DQN models appear to be outperforming lower collision rates than QR-DQN, highlighting the correlation between taking a conservative approach and avoiding collisions, leading to safer highway navigation.

4.3 Discussion

The findings from this experiment confirm previously found performance limitations that DQN suffers from, achieving lower results when the environment setting differs from the training one. This can be seen in related studies [14] and [13], confirming the issues the model has in terms of overestimation bias and lack of performance in out of distribution environments. QR-DQN proves to overcome these limitations by achieving both a higher reward value and lower collision rates, performing better than DQN in most of the scenarios. Overall, the RA QR-DQN variants seem promising for training policies that reduce the probability of collisions, even in environments that differ from the training environment.

Table 2: Average return ± standard error over 5 seeds across different environments.

Model	3 lanes	4 lanes	6 lanes	aggressive	traffic
DQN	30.392 ± 1.010	29.959 ± 0.953	29.339 ± 0.850	29.050 ± 0.971	17.099 ± 0.603
QR-DQN	31.162 ± 0.616	32.637 ± 0.343	32.268 ± 0.392	28.320 ± 0.711	15.647 ± 0.971
RA QR-DQN 0.1	31.400 ± 0.259	31.664 ± 0.207	31.097 ± 0.435	29.606 ± 0.392	15.493 ± 0.791
RA QR-DQN 0.4	31.696 ± 0.165	32.370 ± 0.134	32.057 ± 0.114	28.819 ± 0.420	16.131 ± 0.297

Table 3: Collision rates (%) ± standard error over 5 seeds across different environments.

Model	3 lanes	4 lanes	6 lanes	aggressive	traffic
DQN	20.360 ± 5.914	20.780 ± 5.351	20.800 ± 4.727	24.480 ± 5.272	81.360 ± 1.499
QR-DQN	15.420 ± 3.159	5.220 ± 1.501	5.600 ± 1.689	25.040 ± 3.222	86.800 ± 2.946
RA QR-DQN 0.1	5.720 ± 1.673	4.440 ± 1.902	6.620 ± 3.088	13.040 ± 2.199	84.580 ± 1.579
RA QR-DQN 0.4	8.680 ± 1.399	2.740 ± 0.603	2.600 ± 0.252	20.140 ± 2.379	83.880 ± 0.833

The performance increase between the `3 lanes` and `4 lanes` tests for the QR-DQN models indicate the capacity to which they can adapt to and utilise the added lane for maximising results. However, the decrease in performance from `4 lanes` and `6 lanes` emphasises the limitations of all models to utilise the full available driving space, based on the learned policy. The environments that have higher complexity prove to be a challenge for all models, showcased in the loss of average reward and increase in collisions rates. Surprisingly, the overfitting nature of DQN helps the model achieve better results, as the environments share the same number of lanes as the training one.

From the inconsistencies between the two evaluation metrics, the default reward function offered by HighwayEnv seems to be inefficiently designed. Some models achieve higher reward values, although there is an increase on average collision rates. This is not intuitive, as it seems like the environment reward system allows for high rewards even though the ego vehicle collides, a trend which can be seen throughout multiple environments. In the `3 lanes` environment, this inconsistency is present when comparing the RA QR-DQN models. The `4 lanes` and `6 lanes` tests see QR-DQN achieving a higher reward value despite its higher collision rate when compared to the RA QR-DQN 0.4 model. Likewise, the same behaviour is spotted when comparing QR-DQN to RA QR-DQN 0.1 in the `4 lanes` and `traffic` test environments. These unpredictable results contradict the desired performance, where a conservative approach that achieves lower collision rate should be rewarded higher.

5 Conclusion

This research investigates the robustness of Deep Q-Network (DQN) and Quantile Regression Deep Q-Network (QR-DQN) models in the context of autonomous driving using a simulated highway environment. The primary goals are to provide a risk-averse implementation derived from the standard QR-DQN model and compare the performance of the models in varying environments to assess their robustness.

The findings of this study reinforces results from previous studies and provides significant new insights. The results show the limitations that DQN demonstrates in out of distribution environments, suffering from overestimation bias. This leads to the model performing worse than the QR-DQN counterparts in both the average reward and collision rate. By modeling the distribution over returns using quantiles, the QR-DQN models show improved robustness and lower collision rates when compared to DQN. However, for more complex environments, insufficient training leads the QR-DQN models to perform worse than DQN. An underlying problem concerning the reward function definition is also found, shown by offering high rewards to the agents although they collide, a safety compromise that is not desirable.

The proposed modifications of the QR-DQN models show a risk-averse implementation that utilises the model's quantiles. By considering only the lower part of the quantile, a policy that favours conservative actions can be derived. The changes made involves the model taking a new parameter, named `'quantile_fraction'`, that is used to determine the quantile segment when predicting actions. Considering two models, with `'quantile_fraction'` values 0.1 and 0.4, both Risk-Averse QR-DQN (RA QR-DQN) models outperformed the standard model in terms of collision rates with only minor losses in achieved reward, suggesting an added benefit from incorporating risk-sensitivity in the model's policy. Furthermore, the two models illustrate a critical trade-off between conservative decision-making and reward exploration, showcased by the variation in performance.

Limitations and Future Work

There are some limitations presented by this study. Although the models have been tested in multiple environment variants, the study is limited to only the highway environment, preventing a generalisation of results. Moreover the results found are limited to the model's configuration and training done within the scope of this study, leaving room for better hyperparameter optimisation and more extensive training.

Another limitation in generalisation is determined by the narrow selection of Deep RL models that are being tested, allowing for more advanced models to be studied alongside, for a better understanding of the models' performance.

Multiple open questions are available for further research. A study into optimising the HighwayEnv's highway reward function can be done, in order to find a balance between reward exploration and collision penalty. Furthermore, a study can be conducted to include more than just two RA QR-DQN models, evaluating the trade-off between different `'quantile_fraction'` values. Instead of fixed quantile fraction, future work could explore an implementation that considers adjusting the quantile range dynamically to create more adaptive and context-aware agents.

Responsible Research

The practices of responsible research practices have been considered throughout the process of this study. In this section, an overview of how these practices were implemented is given, touching on the ethical concerns in regards to the study, the reproducible nature of the experiment and how AI has been used in the process.

Reproducibility

Reproducibility was the main goal when designing the experiment and documenting the process. Throughout the experiment implementation phase, every decision was noted down and then included in the report, to ensure it can be reproduced easily. In Section 3, where the methodology is presented, a comprehensive description is given, reporting both the setup and configuration of all components. The implementation of the machine learning models is given, mentioning the hyperparameter settings used. Moreover, the environment configuration is described together with the reward function definition. Due to the inherent random nature of the frameworks and models used, certain seeds have been used and documented to be make the experiment reproducible. Furthermore, the scripts used to run the experiment and its' results were made accessible through the Github repository [19].

Ethical Concerns

Considering that this research is related to the field of autonomous driving, there are some ethical aspects that have become apparent. Specifically, when it comes to the societal adoption of autonomous driving and the environment appropriation of this study. One aspect to be taken into consideration is how does this study fit into society when it is expanded into the real-world. Since the experiment is considering a predefined setting (i.e. highway driving with only automated vehicles), it is hard to consider the results gathered in the context of real-life scenarios, where there is a split between drivers that adopt autonomous vehicles and those that do not. This split can lead to a difference in performance for the autonomous vehicles, given that the surrounding environment can be somewhat unpredictable through the introduction of the human factor. This can be further studied and looked into, to understand better how the transition from simulations to real-life instances can be done.

Another ethical aspect of this study is the representativeness of the environment chosen for the experiment. This research uses a highway simulation environment as the setting in which the models get trained and tested. Throughout the experiment, the configuration is set for at least 3 lanes using a speed range of 14 to 30 m/s (50 to 108 km/h). Although this is representative for most highway settings, it does represent a bias towards highway that have certain lane count and maximum speed limits which may not be representative across the globe (i.e. Germany highways that have no speed limits or countries that have 2 lane highways). Furthermore, a lack of diverse parameters that can be configured in HighwayEnv, may lead to a less comprehensive highway environment setting. This experiment considers only straight highways, with no inclination and perfect road surface and no effect of weather conditions. Consequently, the study may not be applicable in less developed countries where the quality of the road surface varies (i.e. potholes or cracks). In addition, the study may exclude countries with variation in altitudes, where the highway is going uphill or downhill, or countries with diverse weather conditions (i.e. heavy rains, storms, snow or sand storms). Therefore, need of further research that consider a more comprehensive environment that includes these varying aspects of highway driving.

Use of AI

Throughout the process of conducting this study and reporting its findings, LLM tools were used to aid in completing the project. The use of LLMs was limited to help in formatting a LaTeX document, setting up environments and debugging errors and for getting a sense of the direction to move towards when configuring parameters. However, the contents of this experiment and research paper are solely produced and written by the researcher, including the results gathered, sources found and written code for running the experiment. A sample of prompts used in the project is given in appendix A.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Mustafa Celikok, for his crucial guidance, support and encouragement throughout this project. I am also thankful to my responsible professor, Dr. Frans Oliehoek, for proposing this project and providing valuable feedback towards the end of the project. Special thanks to my group members who have worked alongside me, giving insightful feedback and support. I would also like to extend my gratitude and appreciate everyone involved in the Faculty of Computer Science and Engineering, for their knowledge and efforts to teach me their courses and ultimately equipping me to be able to complete this project possible.

References

[1] Darsh Parekh, Nishi Poddar, Aakash Rajpurkar, Manisha Chahal, Neeraj Kumar, Gyanendra Prasad Joshi, and Woong Cho. A Review on Autonomous Vehicles: Progress, Methods and Challenges. *Electronics*,

11(14):2162, January 2022. ISSN 2079-9292. doi: 10.3390/electronics11142162. URL https://www.mdpi.com/2079-9292/11/14/2162. Number: 14 Publisher: Multidisciplinary Digital Publishing Institute.

[2] Jingda Wu, Chao Huang, Hailong Huang, Chen Lv, Yuntong Wang, and Fei-Yue Wang. Recent advances in reinforcement learning-based autonomous driving behavior planning: A survey. *Transportation Research Part C: Emerging Technologies*, 164:104654, July 2024. ISSN 0968-090X. doi: 10.1016/j.trc.2024.104654. URL https://www.sciencedirect.com/science/article/pii/S0968090X2400175X.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 1476-4687. doi: 10.1038/nature14236. URL https://www.nature.com/articles/nature14236. Publisher: Nature Publishing Group.

[4] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep Reinforcement Learning for Autonomous Driving: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6): 4909–4926, June 2022. ISSN 1558-0016. doi: 10.1109/TITS.2021.3054625. URL https://ieeexplore.ieee.org/abstract/document/9351818.

[5] Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), March 2016. ISSN 2374-3468. doi: 10.1609/aaai.v30i1.10295. URL https://ojs.aaai.org/index.php/AAAI/article/view/10295. Number: 1.

[6] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

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

[8] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992698. URL https://doi.org/10.1007/BF00992698.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[10] Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, March 1964. ISSN 0003-4851, 2168-8990. doi: 10.1214/aoms/1177703732. URL https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-35/issue-1/Robust-Estimation-of-a-Location-Parameter/10.1214/aoms/1177703732.full. Publisher: Institute of Mathematical Statistics.

[11] T Tieleman and G Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4:26–31, 2012.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. URL http://arxiv.org/abs/1412.6980. arXiv:1412.6980 [cs].

[13] Marcell Adi Setiawan, De Rosal Ignatius Moses Setiadi, Erna Zuni Astuti, T. Sutojo, and Noor Ageng Setiyanto. Exploring Deep Q-Network for Autonomous Driving Simulation Across Different Driving Modes. *Journal of Future Artificial Intelligence and Technologies*, 1(3):217–227, October 2024. ISSN 3048-3719. doi: 10.62411/faith.3048-3719-31. URL https://faith.futuretechsci.org/index.php/FAITH/article/view/31. Number: 3.

[14] Sandy Nugroho, De Rosal Ignatius Moses Setiadi, and Hussain Md Mehedul Islam. Exploring DQN-Based Reinforcement Learning in Autonomous Highway Navigation Performance Under High-Traffic Conditions. *Journal of Computing Theories and Applications*, 1(3):274–286, February 2024. ISSN 3024-9104. doi: 10.62411/jcta.9929. URL https://publikasi.dinus.ac.id/index.php/jcta/article/view/9929. Number: 3.

[15] Esther Aboyeji, Oladayo S. Ajani, and Rammohan Mallipeddi. Effect of Number of Lanes on Traffic Characteristics of Reinforcement Learning Based Autonomous Driving. *IEEE Access*, 11:80199–80206, 2023. ISSN 2169-3536. doi: 10.1109/ACCESS.2023.3299860. URL https://ieeexplore.ieee.org/abstract/document/10196379.

[16] Rishabh Sharma and Prateek Garg. Optimizing Autonomous Driving with Advanced Reinforcement Learning: Evaluating DQN and PPO. In *2024 5th International Conference on Smart Electronics and Communication (ICOSEC)*, pages 910–914, September 2024. doi: 10.1109/ICOSEC61587.2024.10722344. URL https://ieeexplore.ieee.org/abstract/document/10722344.

[17] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16. PMLR, October 2017. URL https://proceedings.mlr.press/v78/dosovitskiy17a.html. ISSN: 2640-3498.

[18] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit Quantile Networks for Distributional Reinforcement Learning, June 2018. URL http://arxiv.org/abs/1806.06923. arXiv:1806.06923 [cs].

[19] Cristian Toadere. Research project experiment code-

base, found in the rp_cristian directory. https://github.com/Tdr13/research_project_25, 2025.

[20] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

A LLM prompts used in the research project

Latex Formatting

- https://chatgpt.com/share/685160cf-4c84-800c-bfe1-fea5d9cb6c85
  - "How to use citet in latex?"
  - "Does it work if there are more authors?"
  - "I want to put 6 figures in 2 rows and 3 columns in latex. How do i do that, given that i am using a 2 column format?"
  - "how to put svg in latex?"
  - "Package svg Error: File dqn_length_svg-tex.pdf' is missing. Package svg Error: File dqn_length_svg-tex.pdf_tex' is missing."
- https://chatgpt.com/share/6851611d-2c48-800c-bbf3-24740f09bd8f
  - "Can you create a table in Latex using this? I want the headers to be formatted as bold and I want separating lines only between the headers and body"
  - "My table is in a column of a two column document, but the span of the table goes on both columns"
  - "how to deliimit the right margin of the table, it goes over the page limit"
  - "can i put distance between the rows of the table?"
- https://chatgpt.com/share/6817a93d-1e30-800c-830e-2713aba4168a
  - "Underfull hbox (badness 10000) in paragraph at lines 2–4 how to solve in latex"
  - "how to make latex text bigger"
  - "How to make the latex citations appear in order in the text"
  - "how to define the width for just one page in latex"
  - "How to set size of title using the ijcai style"

Setup and Debugging

- https://chatgpt.com/share/68516101-5d60-800c-a799-3635c9e8086c
  - "(base) PS C:\TU Delft\RP\rp_cristian git push origin main error: src refspec main does not match any error: failed to push some refs to 'github.com:Tdr13/research_project_25.git'"
  - "no branch appears when git branch"
  - "how to ignore the .idea folder in my project?"
- https://chatgpt.com/share/685163e7-4f08-800c-b221-fe35fcbf3ce5
  - "How to setup highwayenv in conda?"
- https://chatgpt.com/share/68516594-f6ac-800c-ada8-ea77932fa3c6
  - "I get this error when trying to run a pyhton file using py in conda: C:\Users\nikev\AppData\Local\Programs\Python\Python312\python.exe: can't open file 'C:\\TU Delft\\RP\\highway-rp \\main_qrdqn.py': [Errno 2] No such file or directory"
- https://chatgpt.com/share/685166bd-fdac-800c-b883-f35d1892c44d
  - "how to make a stable baselines 3 model that is being trained in the highwayenv run on my gpu instead of my cpu"

Questions

- https://chatgpt.com/share/685160cf-4c84-800c-bfe1-fea5d9cb6c85
  - "What does disableing benchmaring ad enforcing determinstic behaviour in pytorch mean?" and "what does the off-policy nature of an rl algorithm means?"
- https://chatgpt.com/share/685163ae-55d8-800c-8559-c86fc98f9dd5
  - "How to seyup the pythonhashseed environment variable?"
  - "what about within the python file?"
- https://chatgpt.com/share/68516163-0aa4-800c-96f5-68426c65dff1
  - "What is the simulation_frequency of highwayenv mean?"
- https://chatgpt.com/share/6810fd67-45e0-800c-ae80-684b21ebe9fd
  - "how many timesteps should I use when trining DQN?"
  - "What if I am using highwayenv"