Optimizing driving entity switching of semi-automated vehicles under automation degradation

Csanád Bakos
Technical University of Delft, Netherlands
Faculty of Electrical Engineering, Mathematics & Computer Science

Abstract

Transitioning to use automated vehicles is a gradual process. Until full automation capabilities are developed there is a need to mediate which driving entity - human or autonomous driving system (ADS) - should be in control depending on the circumstances. This research aims at investigating the switching between manual and automated driving in semi-autonomous vehicles when the ADS becomes unfit to drive. To this end, a simple environment simulation was created and an MDP model was formulated that accounts for sensor failures and leaving the operational design domain (ODD). Deep Q-Network (DQN), a deep reinforcement learning (RL) algorithm was trained and evaluated against a hand-curated decision-tree-based standard. The DQN-based policy did not reach the performance of the baseline algorithm. The conclusion is drawn that using DQN to handle this multi-objective decision problem using an intuition-based reward function cannot learn an optimal policy.

1 Introduction

Automated driving systems have seen much attention and development in recent years. These technological advancements come with great safety potential, however in some cases the human driver needs to take over. This is caused by the automated system not (yet) supporting all possible scenarios that can occur on the road.

MEDIATOR [1], funded by EU Horizon 2020, aims to develop a mediating system that enables a safe and real-time switching between the human driver and the automated system by continuously assessing the driving fitness of both in various driving contexts. The Mediator system is generally subdivided into five main modules, including the driver module, the automation module, the context module, the human-machine interface (HMI) module, and the decision logic module [1, 2].

In this study, the focus is on the Decision Logic module. Its functionality is concerned with determining the actions regarding the transfer of control to guarantee both safety and driver comfort. During this process, the decision logic module receives inputs from the driver and automation module and selects appropriate actions based on information about past, current and expected situations. The selected actions are then sent to either the HMI or the automation module that is responsible for the execution and monitoring of the actions [2]. In this project, two automation levels are used: L0 (manual mode / no automation) and L4 (high automation) in order to limit complexity.

There is existing research investigating optimal switching policies between manual and automated driving in semi-autonomous vehicles. The decision problem can be modeled by a Markov Decision Process (MDP) or Partially Observable MDP (POMDP) formulation under full and partial information [3, 4]. In the latter approach, for example a deep RL algorithm, (Asynchronous Advantage Actor Critic) A3C [5] can be used to obtain an approximate solution [3]. These studies are optimized over simulated environments which need further fine-tuning to allow better generalization of the learned policies to the real world [3, 4].

Several studies have been investigating the human factor of switching between driving entities [6, 7, 8, 9]. Self-monitoring to ensure that the automated system is fit to drive also has been studied [10, 11, 12]. The Operational Design Domain (ODD) describes the specific operating domains in which the automated driving system (ADS) is designed to function [13]. The ODD is expected to be different for each ADS feature on a vehicle and should specify the condition in which that feature is intended to operate. Factors that influence this are roadway types, speed range, lighting conditions, weather conditions, and other operational constraints [13]. A key observation is that the ADS should be able to detect when it is within or outside of its ODD [13].

There are a number of things that are missing from existing models. Firstly, one can construct a belief about the time it takes for the driver to take over the vehicle control from the car, instead of using a priori given value [3]. Secondly, the action space used in the model can be expanded to allow emergency stops and switching within different automation levels. Furthermore, the representation of the ADS being (un)fit to drive can be improved by adding support for intrinsic states, thus being able to account for faults of the autonomous driving entity. Finally, modelling driving entity switches due to leaving the ODD is a yet to be investigated topic.

This paper is concerned with the following **research question**: how to compute an optimal policy for switching between driving entities when the automation becomes unfit to drive?

To answer the main research question, the following sub-questions are addressed:

Q1: How can the decision problem formulated as MDP and what are the state space, action space, transition function, and reward function?

Q2: How to simulate observations defined in the MDP state space to reflect the reality of sensor failures and entering/leaving ODD?

Q3: How do deep reinforcement learning methods compare to a hand-curated decision tree policy in terms of safety and comfort?

In the next section, - methodology - firstly an outline is given regarding how the research questions are addressed in the paper. The following part is concerned with the MDP formulation and the simulation of the state space variables. Next the experiments, including the results are dissected. In the responsible research part, the relevant ethical concerns are on the table. Finally, conclusions are drawn together with an outlook on future work.

2 Methodology

In this section an overview is given regarding how the research questions are addressed in the paper.

Firstly, an MDP is formulated, which is known to be suited to define stochastic control problems [14] - such as this use case. Once defined, different methods can be used to compute a policy, which in turn enables one to select what action to execute as a function of state [14]. The suitability of using MDPs is supported by previous papers in this area of research as well [3, 4]. A common technique for computing policies is to use reinforcement learning.

In general, there are two possibilities to use RL. One can use it directly - e.g. Atari games - where the agent can directly interact with its environment repeatedly without any issues. In many cases, this is not possible because the cost is too high. For instance, throwing away thousands of self driving cars just for training an RL agent for switching between driving entities is not feasible. In order to compute optimal switching policies, one must first construct a simulation environment that a decision agent can interact with. Once the simulation is made realistic enough and an agent mastered its task in it, a future challenge will be to bridge the gap between the simulation and reality which is often a difficult task. Optimizing driving entity switching under automation degradation is in an early stage of research which justifies making a simplified simulation.

The first step in designing the simulation environment is to formulate the state space of the MDP model which prescribes the observations that must be simulated. The MDP formulation provides all the necessary parts for using RL.

After creating the MDP formulation and implementing the simulation, different algorithms can be used for optimizing driving entity switching. In this study, to establish a baseline, a decision-tree based policy is implemented. Also, different versions of DQN are used to see how deep RL performs on this task. Using deep RL instead of traditional RL methods like Policy-iteration is motivated by having a high dimensional action-state space.

Finally, various metrics are defined and tracked in order to assess the safety and comfort of the obtained policies by the different methods.

3 Modeling for optimal control switching

In this section, the modeling of the decision problem is addressed step-by-step. Firstly, creating the MDP formulation is discussed, followed by setting up the simulation environment.

3.1 Markov Decision Process

MDPs work under full information, therefore the assumption is made that observations are completely reliable. In real life settings, this is a strong assumption, and POMDPs offer an alternative model to account for unreliability at the cost of higher complexity.

Next, the four elements of an MDP are defined for this use case.

State space This prescribes which observations of the environment the mediator agent has access to at each time step.

For this use case, three aspects were identified about the human driver for the observations. First, when in L4, the system needs to know how quickly the human can take back control once alerted. Secondly, in L0, an estimate is needed regarding how long can the person drive safely (e.g. due to getting tired). Finally, the person needs to have authority over the decision whether the driving control switch takes place. The assumption is made that the person does not initiate take overs, only the mediator system does.

The following observations were designed for these purposes about the human driver:

- **TTDF**: time to driver fitness - number of time steps until the person can take over - used in L4
- **TTDU**: time to driver unfitness - number of time steps until the person can no longer drive - used in L0
- **DriverResponse**: Response of the person to the suggested action by the mediator system (accept, reject or no response)
- **NDRT**: Non-driving related task - when ADS is in control, TTDF is influenced by what activity (e.g. reading, texting) the person is engaged in.

For the autonomous system, it is assumed that the system can predict the time until it enters or leaves its ODD. Automation system failures are assumed to be unpredictable, however, are detected at the the time step when they occur. Also, knowing the current and highest available automation modes were identified as relevant observations.

The following were designed to represent the autonomous system in accordance with the above defined assumptions.

- **LevelMaxNow**: maximum automation level available at a given time step
- **AutomationMode**: current automation level

- **SF4**: (automation) system failure at level 4 (true or false), if it is true then the given level is not available - models sudden, unpredictable failures that occur for the automation system due to sensor failures, lane markings not visible, encountering a construction site that was unknown and possibly more.
- **TTA4U**: time to leave ODD/ time to automation unfitness for automation level 4
- **TTA4F**: time to enter ODD/ time to automation fitness for automation level 4

In order to address the different risk levels of driving entities being in control of the car, given different scenarios, the following two, discretized variables were added, each with 3 levels: low, medium and high risk.

- **HR**: human risk - determines the probability of an accident at the given time step, assuming the person is in control.
- **A4R**: autonomous-system risk for level 4 - determines the probability of an accident at the given time step, assuming the ADS is in control.

To conclude about the state space, Table 1 gives an overview regarding the values and the corresponding meanings of the observations.

Observation	**Range**	**Values**
TTDF	[0, 9999]	0: human is fit, x: x time steps until human is fit
TTDU	[0, 9999]	0: human is unfit, y: y time steps until human is unfit
DriverResponse	[0, 2]	0: accept, 1: reject, 2: no response
NDRT	[0, 2]	0: none, driver is alert, 1: obstruction, 2: immersion
LevelMaxNow	[0, 1]	0: L0, 1: L4
AutomationMode	[0, 1]	0: L0, 1: L4
SF4	[0, 1]	0: no L4 ADS failure, 1: L4 ADS failure occurs
TTA4F	[0, 9999]	0: L4 ADS is fit, w: w time steps until L4 ADS can start driving
TTA4U	[0, 9999]	0: L4 ADS is unfit, z: z time steps until L4 ADS can still drive
HR	[0, 2]	0: low, 1: moderate, 2: high human risk level
A4R	[0, 2]	0: low, 1: moderate, 2: high L4 ADS risk level

Table 1: Overview of observations and their values

Action space The action space is discrete. The action transition probabilities are 1, thus when the agent picks an action, then that is carried out with full certainty.

- DN: Do Nothing - let the current entity in control keep driving
- SL0: suggest a shift to L0, includes preparing the driver for the shift
- SL4: suggest a shift to L4, includes preparing the driver for the shift
- EL0: enforce a shift to L0, give control to the human
- EL4: enforce a shift to L4, give control to the ADS at L4
- ES: Emergency stop - stop the car by the side of the road

Reward function In this section, firstly an overview is given about the desired traits that the model should learn. Afterwards the different reward setups that were evolved are showcased.

Learning objectives

- **O1**: Do emergency stops when both driving entities become unfit or a shift is not possible to a safe level (e.g. due to not getting response from human driver in time)
- **O2**: Minimize accidents
- **O3**: Learn suggesting shifts when driving entity is reaching unfitness and then enforcing these once approved by the driver
- **O4**: Avoid unnecessary actions (e.g. suggest shifts when not needed) for driving comfort

The reward function was built in multiple steps. In the beginning, rewards were kept simple; accidents and being in higher risk were penalized while reaching the end of the sequence and driving in low risk was rewarded. Figure 1 illustrates this.

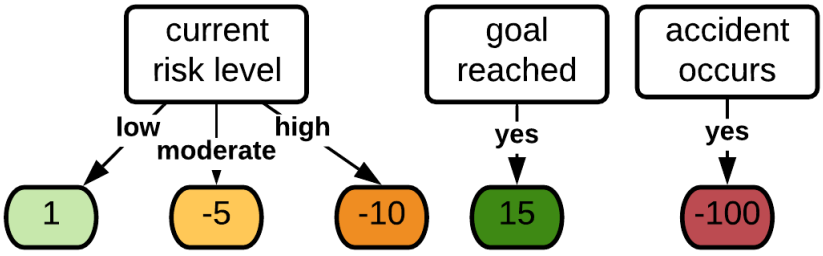


Figure 1: Base rewards

After some initial trials, it became clear that the agent cannot learn getting approval from the driver for shifts (O2) this way. Another issue arise from having a lot of unnecessary actions making the ride uncomfortable (O4). The decision was made to add further rewards to help the agent correct its behaviour. Figure 2 gives the details of the additional rewards that were added on top of the previously illustrated ones as seen in Figure 1.

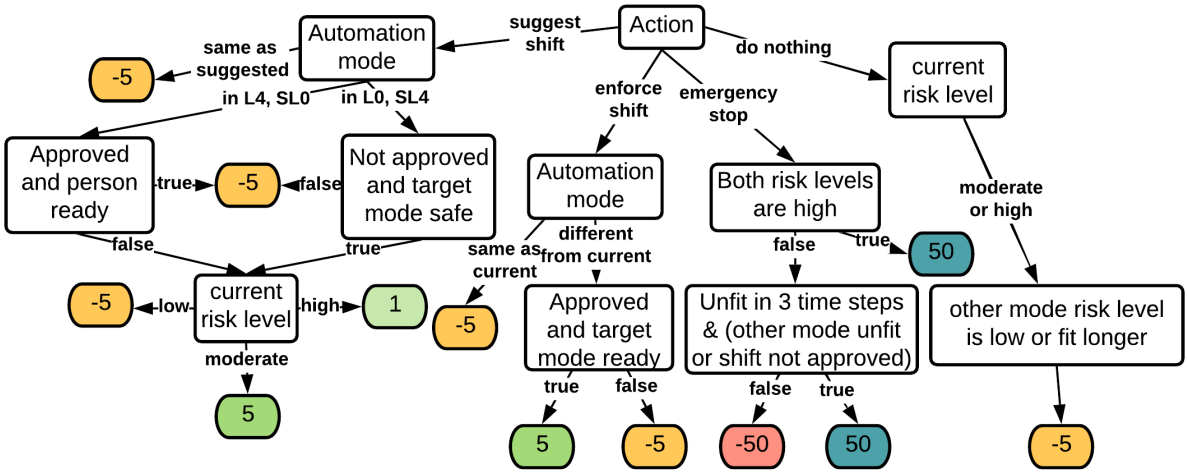


Figure 2: Additional rewards depending on observations after taking an action

State transition function For some applications, this can be directly defined, however, in most applications - including this use case - the state transition function is estimated by letting the agent interact with the environment. Furthermore, due to using deep reinforcement learning, the model learns the transition probabilities during training, thus we don't compute this explicitly.

3.2 Environment simulation

In this section, firstly, the simulation of the MDP state space variables is described. Afterwards, more details follow on how the different parts of the simulation fit together.

There are two main parts. Firstly, there are variables that can be computed before the episode starts thanks to their independence from the agent's actions. These are stored in a table and looked up during the simulated runs. The second part consists of observations that are influenced by the agent's actions, these are computed on the go. Finally, there are variables in between, that are pre-computed but in some cases need to be recomputed during the simulation.

Pre-computed observations

LevelMaxNow: In order to have situations where the agent runs out of its ODD, possibly repeatedly, 1-5 random time steps are assigned (per episode) as switch times, where the level max is randomly assigned to be L0 or L4. This random assignment happens at the first time step as well. Going through the episode step by step, when a switch time is hit, the newly chosen level is assigned repeatedly until the next switch time is reached. Then the cycle starts again. This way L4 is available for 20-50 time step parts on average in one episode.

TTA4F: Computed based on LevelMaxNow. At each time step its value is zero if L4 is available. If only L0 is available at that point then the number of time steps until L4 is available again is assigned. In case L4 won't be available anymore in the episode, then assign tt_max (set to 9999).

TTA4U: Computed based on LevelMaxNow. At each time step its value is zero if only L0 is available. If L4 is available at that point then the number of time steps until only L0 is available again is assigned. In case L4 will be available for every time step in the rest of the episode, then assign tt_max (set to 9999).

SF4: First, it is determined whether this episode will have a SF4 at all (set to happen with 0.5 probability). By default, all time steps have no SF4. If a system failure is to happen, then a random time step is chosen within the sequence where it occurs. All time steps after the first occurrence are also set to have SF4 in the episode.

Recompute rules are needed because the assumption is made that we cannot predict SF4 in advance. If we would take into account SF4 while pre-computing LevelMaxNow, TTA4U and TTA4F then this assumption would not hold. However, the agent relies on these observations to be correct, therefore, once we hit a case of SF4, these need to be adjusted.

LevelMaxNow: when the first time step where SF4 occurs is reached, LevelMaxNow is set to L0 for the rest of the sequence.

TTA4F: when the first time step where SF4 occurs is reached, TTA4F is set to tt_max (9999) for the rest of the sequence.

TTA4U: when the first time step where SF4 occurs is reached, TTA4U is set to 0 for the rest of the sequence.

On-the-go observation simulations The following observations are computed at each time step: NDRT, TTDF, TTDU, HR, A4R, AutomationMode and DriverResponse. These cannot be pre-computed because of their dependence on the agent's actions.

AutomationMode: for the first time step, it is randomly chosen between L0 and L4. Afterwards, it is solely determined by the actions of the agent. Only EL0 and EL4 can alter this state.

DriverResponse: By default, it is set to 'no response'. When SL0 or SL4 action is taken, then it is set to 'accept', 'reject' or 'no response' according to the configured probabilities. For the experiments, these probabilities were set to 0.8, 0.1 and 0.1, respectively. This models the case of a relatively agreeable driver because it was assumed that the agent can persuade the driver to take the suggested actions.

NDRT: When in L0, the driver cannot engage in any activities other than driving, thus set to level 0 - none. An exception to this, is the edge case when EL0 is taken before TTDF is zero, then it may or may not be true that NDRT is none. (The driver might be still reading for instance, or the driver finished her NDRT earlier than expected) Either way, the risk level calculation accounts for this based on the TTDF observation. For more details, see the TTDF simulation.

The other possibility is that we are in L4. For this simulation, it is assumed that the driver does not change her NDRT within one L4 part of the sequence because of the short L4 lengths. Thus, an NDRT level of 1 or 2 is assigned when switching to L4 and otherwise the previous NDRT level is kept. See Figure 3 as well for more insight.

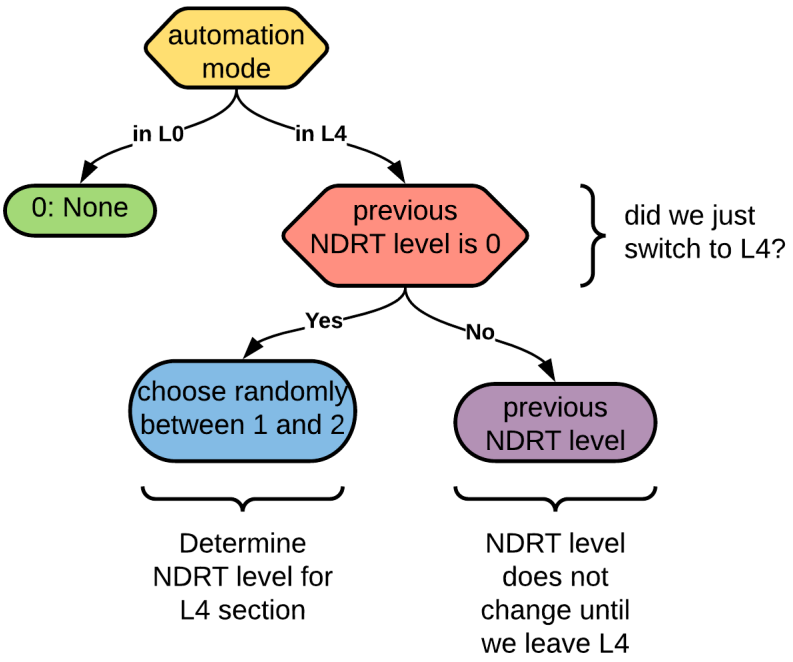


Figure 3: Simulating NDRT

TTDF: In L0, ideally it is zero, however if a shift was enforced too early then TTDF might not be zero yet. In L4, if NDRT is 0, meaning that the driver is alert, then TTDF is 0. Otherwise depending on the NDRT level, TTDF is either 5 or 10 time steps. TTDF can be reduced only by taking the action SL0, which prepares the driver for a shift to manual control.

TTDU: It is determined depending on the previous TTDU value and the current automation mode. When entering L0 (by EL0 action in L4) TTDU is randomly assigned from the range of 30 to 50 time steps. Then, while in L0, TTDU is reduced by one at each time step, until hitting zero. When in L4, TTDU is set to zero, because NDRT is always level 1 or 2 in the simulation, meaning that TTDF is never zero, unless a shift is initiated through SL0.

(a) Simulating TTDF

(b) Simulating TTDU

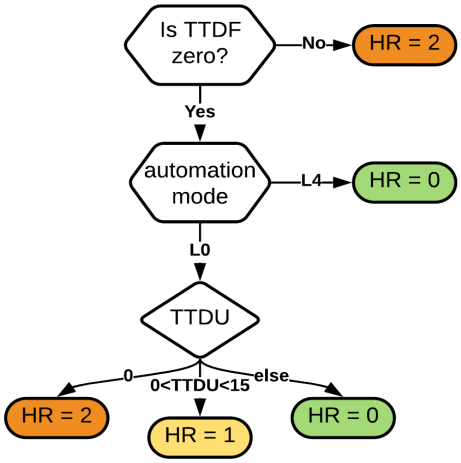
Figure 4: TTDF and TTDU

HR: High risk when TTDF is not zero. In L4, becomes low risk only once TTDF is zero. When in L0, it is set to zero (low risk) as long as the human driver is still fit to drive at least 15 time steps. Below 15 time steps, the risk level is moderate until TTDU is zero. Human risk levels are simulated this way to reflect that letting the person to be in control before she is ready or after she became unfit, is dangerous. The moderate risk level was created to signal reaching high risk levels.

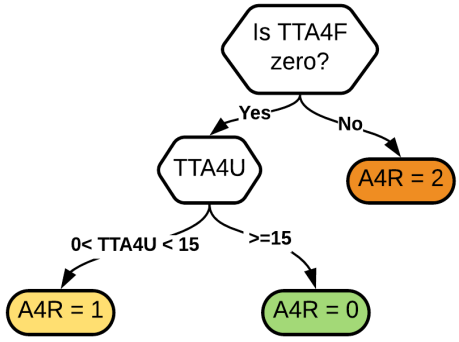
A4R: When TTA4F is not zero, the L4 ADS is unfit, thus A4R is set to high risk. When L4 ADS is fit for at least 15 more time steps then the risk is low, once below 15 time steps means moderate risk level, given that TTA4F is still zero. Autonomous system risk was designed with the same objectives in mind as HR, differing only in terms of taking the ADS into account instead of the human driver.

Accidents are a consequence of letting an unfit entity keep driving. When the human driver becomes unfit, the HR observation reflects this by being set to high risk. Similarly, if the L4 ADS is unfit - due to being out of ODD or having a sensor/system failure - then A4R signals to the agent that driving in L4 has a high risk. These risk levels determine the probability of an accident happening. In particular, the probability of accident in high risk level is 0.1 at each time step. Thus, the agent can expect an accident by ten time steps in high risk level with overwhelming likelihood. Low risk level was set to 0 accident probability for simplicity. Also, in moderate risk level accidents cannot occur because this only meant to signal that a high risk level is around the corner.

Overall simulation Each episode is simulated by firstly computing the pre-determined observations and setting all the observation values for the first time step. Afterwards, at each step, the on-the-go observations are computed based on the action taken by the agent. Also, it is checked if any pre-computed values need to be reevaluated. Once the environment observation values are ready, they are available to the agent through the interface of the OpenAI Gym environment [15].



(a) Simulating HR



(b) Simulating A4R

Figure 5: HR and A4R

4 Experiments

This section discusses the experiments that were done in order to compute and evaluate switching policies for the automation degradation use case. First the used algorithms are described followed by defining evaluation metrics. The section concludes by reporting the performance of the found policies according to the evaluation metrics.

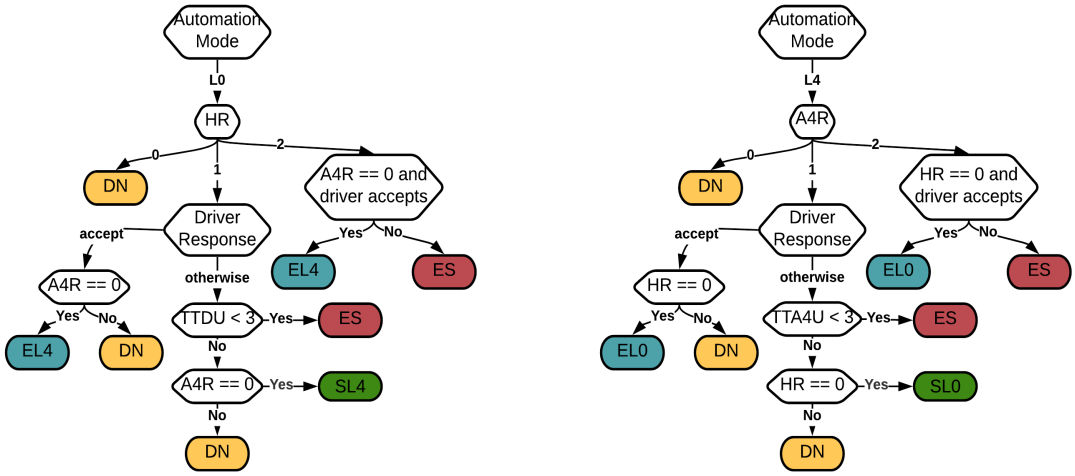
4.1 Decision-tree-based baseline policy

A baseline policy was designed in the form of a decision tree, in order to establish a base performance. The algorithm is optimized for safety and driver comfort. For safety, the algorithm suggests and enforces shifts only to low risk levels. When this is not possible and the current driving entity is becoming unfit, an emergency stop is carried out. For comfort, suggestions to shift are only made when driving entity is becoming unfit, and enforcing shifts are only done after approval from the human driver. For more insight, see Figure 6a and 6b.

4.2 Deep Q-Network

In order to assess the performance of deep reinforcement learning for this use case, a base implementation of DQN from Stable Baselines 3 (SB3) was used [16]. DQN was chosen because it is a classic algorithm, that has proven itself already across several tasks. Unfortunately, there are some extensions to DQN which are not (yet) included in the SB3 implementation. To this end, the SB1 [17] DQN implementation was also utilized which contains the prioritized replay and dueling extensions as well.

Hyper-parameter optimization While the simulation was being developed, experiment tracking was setup early using Weights & Biases (WandB) [18]. This enabled using the 'Sweeps' hyper-parameter optimization functionality from WandB. The Bayesian search method was chosen which relies on a Gaussian Process to model the relationship between the parameters and the model metric, which was set to be the (mean) episode length. Each run consisted of 1 million training episodes. See Figure 7 in the appendix for the hyper-parameter optimization results. The details of the sweep configuration will be made available together with the source code of the project upon request.



(a) In L0

(b) In L4

Figure 6: Baseline decision-tree-based policy

Analyzing the best episodes, the hyper-parameters were set for the experiments to be described later on. This way, the hyper-parameters of DQN were setup to work supposedly well on the designed observation space.

Training During training, the mean episode length and mean reward across the episodes were tracked and plotted via WandB. These metrics helped to check if the training converged. However these provided no information regarding the correctness of the computed policy. To this end, this process consisted of training, running evaluation, ending with adjusting rewards and parts of the simulation, then start again with training.

4.3 Evaluation metrics

To evaluate the policies, a number of metrics were used. In this section, the evaluation metrics are described. The main evaluation categories are driving safety, driving comfort, decision efficiency and decision accuracy.

Driving safety

1. Ratio of episodes ending with accident (smaller is better), computed according to Eq. (1).

$$\textbf{Accident ratio} = \frac{\text{number of episodes ending with accident}}{\text{total number of episodes}} \tag{1}$$

2. Ratio of all time steps in HR level 2 - high risk - while human in control, (smaller is better), computed according to Eq. (2).

$$\textbf{HR level 2 ratio} = \frac{\text{number of time steps across all episodes when HR} == 2 \text{ and in L0}}{\text{total number of time steps across all episodes}} \tag{2}$$

3. Ratio of all time steps in A4R level 2 - high risk - while ADS in control, (smaller is better), computed analogously to Eq. (2) using A4R and L4.

4. Ratio of unsafe actions, (smaller is better), includes:

- SL4 in L0 when A4R == 2
- EL4 in L0 when A4R == 2
- SL0 in L4 when HR == 2
- EL0 in L4 when HR == 2

Computed according to Eq. (3).

$$\textbf{Unsafe action ratio} = \frac{\text{number of time steps across all episodes when an unsafe action is taken}}{\text{total number of actions taken across all episodes}} \tag{3}$$

Driving comfort

1. Ratio of unnecessary actions, includes the following scenarios:

   - SL0 in L0
   - SL4 in L4
   - EL0 in L0
   - EL4 in L4
   - SL0 in L4, when A4R=0
   - SL4 in L0, when HR=0
   - EL0 in L4, when A4R=0
   - EL4 in L0, when HR=0
   - ES in L4, when A4R=0
   - ES in L0, when HR=0

   Computed according to Eq. (4).

   Unnecessary actions ratio =

   $$\frac{\text{number of time steps across all episodes when an unnecessary action is taken}}{\text{total number of actions taken across all episodes}} \tag{4}$$

2. Ratio of episodes ending with reaching end of sequence, (larger is better), computed according to Eq. (5).

   Complete episode used ratio =

   $$\frac{\text{number of episodes ending with reaching end of the sequence}}{\text{total number of episodes}} \tag{5}$$

3. Ratio of approved enforced shifts to L0 (larger is better), computed according to Eq. (6).

   Approved EL0 ratio =

   $$\frac{\text{number of EL0 actions taken with approval from driver across all time steps}}{\text{total number of EL0 actions taken across all time steps}} \tag{6}$$

4. Ratio of approved enforced shifts to L4 (larger is better), computed analogously to Eq. (6) using EL4.

Decision efficiency Measured by **decision duration**. A duration starts when entering risk level 1, and ends when a shift is enforced or emergency stop is executed. Suggesting shifts is an intermediate step, therefore those are not taken into account. However, in order to account for too late decisions, ending with accidents are added to this average as well. When the end of the sequence is reached, the duration count is reset (if it was activated) without adding the length to the recorded ones. The mean, standard deviation (SD), median, min and max values of the decision duration are computed.

Decision accuracy

1. Ratio of enforced shifts from L4 to L0 when A4R > 0 and HR $== 0$, (larger is better), computed according to Eq. (7).

$$\textbf{Needed EL0 action ratio} = \frac{\text{number of needed EL0 actions across all time steps}}{\text{total number of EL0 actions taken across all time steps}} \tag{7}$$

2. Ratio of enforced shifts from L0 to L4 when HR > 0 and A4R $== 0$, (larger is better), computed analogously to Eq. (7).

3. Ratio of needed SL0 actions, SL0 is needed when in L4, A4R > 0 and either HR > 0 or driver is not (yet) accepting the shift, computed analogously to Eq. (7).

4. Ratio of needed SL4 actions, SL4 is needed when in L0, HR > 0 and A4R $== 0$ and the driver is not (yet) accepting the shift, computed analogously to Eq. (7).

5. Ratio of needed and approved enforced shifts to L0 (larger is better), computed analogously to Eq. (7).

6. Ratio of needed and approved enforced shifts to L4 (larger is better), computed analogously to Eq. (7).

7. Ratio of correct emergency stops, (higher is better), correct emergency stops are:

   - Both human and autonomous system is unfit to drive (A4R > 1 and HR > 1)
   - In L4, TTA4U < 3 (almost unfit), driver does not approve shift (no response or reject)
   - In L0, TTDU < 3 (almost unfit), driver does not approve shift (no response or reject)

   Computed according to Eq. (8).

$$\textbf{Correct ES ratio} = \frac{\text{number of time steps across all episodes when a correct ES is executed}}{\text{total number of emergency stops taken across all episodes}} \tag{8}$$

4.4 Results

In this section, the performance of the baseline decision-tree policy is compared with that of the SB3 and SB1 implementations of the DQN algorithm. The earlier defined evaluation metrics are used for this comparison.

To accumulate the results, each policy was used for 5000 episodes. The reward function in the SB3 experiment was slightly altered compared to the SB1 setup, which was illustrated earlier. The WandB experiment id allows better reproducibility because it links the results to configuration files that were saved and are made available upon request.

Driving safety In Table 2 the safety performance of the different policies are assessed. It is clear that the DQN approach lags behind the baseline, although the SB1 version is often comparable to the standard.

Policy (WandB id)	Accident ratio	HR level 2 ratio	A4R level 2 ratio	Unsafe action ratio
Baseline (333-eval)	**0.0128**	**0**	**0.0082**	**0.04632**
SB3 DQN (167-train)	0.1788	0.03868	0.01535	0.1737
SB1 DQN (184-train)	0.026	0.005363	0.02005	0.09648

Table 2: Driving safety results

Driving comfort In terms of driving comfort, looking at Table 3, the baseline performs well, while the deep learning based approaches fail, especially in terms of getting shifts approved before enforcing them.

Policy (WandB id)	Unnecessary actions ratio	Complete episode used ratio	Approved EL0 ratio	Approved EL4 ratio
Baseline (333-eval)	**0**	**0.1762**	**1**	**1**
SB3 DQN (167-train)	0.0851	0.064	0.2231	0
SB1 DQN (184-train)	0.04752	0.0008	0.5798	0.316

Table 3: Driving comfort results

Decision efficiency For decision efficiency, as seen in Table 4, overall the SB1 DQN outperforms the baseline and the SB3 version by making decisions quicker.

Policy (WandB id)	Max	Min	Mean	Median	SD
Baseline (333-eval)	**13**	0	5.99	5	5.229
SB3 DQN (167-train)	99	0	7.187	3	8.773
SB1 DQN (184-train)	14	0	**2.193**	**0**	**3.777**

Table 4: Decision length results

Decision accuracy As seen in Table 5, the DQN based approaches couldn't learn how to suggest and enforce shifts appropriately. The SB1 setup makes some progress in terms of switching to L0 correctly but it is not comparable to the baseline nevertheless. Table 6 gives more insight regarding the approval ratio of the needed shifts. It also exposes the situation regarding the correctness of doing emergency stops. The SB3 configuration works quite well in this case but not quite comparable to the baseline performance.

Policy (WandB id)	Needed SL0 action ratio	Needed SL4 action ratio	Needed EL0 action ratio	Needed EL4 action ratio
Baseline (333-eval)	**1**	**1**	**1**	**1**
SB3 DQN (167-train	0.6287	null	0.2054	0.3262
SB1 DQN (184-train)	**1**	0.3112	0.5798	0.1222

Table 5: Needed shift performance

Policy (WandB id)	Needed & approved EL0 action ratio	Needed & approved EL4 action ratio	Correct ES ratio
Baseline (333-eval)	**1**	**1**	**1**
SB3 DQN (167-train)	0.1982	0	0.8067
SB1 DQN (184-train)	0.5798	0.1222	0.5688

Table 6: Emergency stop and enforced shift correctness results

Overall DQN gives worse performance than the baseline algorithm. This could be an issue due to the reward function, even though more than 200 different training experiments were done trying different reward setups. More details regarding this are given in the discussion section.

5 Responsible research

In this section, the ethical aspects of the research are addressed. This includes reproducibility concerns and the open discussion regarding whether the person or the ADS should have the final word when shifting between automation levels.

In order to make this paper reproducible, a number of steps were taken. Firstly, WandB was used to track all experiments, both during training and evaluation. Experiment tracking included storing all the used hyper-parameters, environment parameters such as episode length and probability functions for driver response and the conditional accident probability depending on the risk level. Furthermore, when doing hyper-parameter optimization, the corresponding configuration files were also saved. All these log data together with the source code of the project will be made available upon request.

While designing the baseline policy and the reward function it was inevitable to touch upon the question whether the human or the autonomous system should have the last word when making shifting or emergency stop decisions. It is relatively easy to come up with scenarios where one or the other is more preferable.

For instance, if we leave the autonomous system in charge, consider the situation from this use case when the ADS fails to detect that the driver is fit. Then the ADS can refuse to give back the control to the human due to its unfit belief about the driver. This is clearly problematic.

On the other hand, if the human has to have the last word in shifting decisions, other issues arise. When the vehicle is in L4, then the person cannot be expected to be responsive in a short interval of time. However, detecting an autonomous system failure can happen from one second to the other. Depending on the severity of the system issue, it might be dangerous to keep driving in L4 even for a few more seconds. Thus, in this case, it is desirable for the system to execute an emergency stop, without approval from the person.

For this project, the compromise was made to do emergency stops without human approval, while shifting between L0 and L4 are oriented towards using human approvals first. This is an open discussion, it seems that some sort of hybrid approach could work. In order to see how exactly, more research will be needed.

6 Discussion

In this section the results are discussed and the known limitations are addressed. Also, the pros and cons of the baseline policy and using DQN are pointed out. Finally, some thoughts on the simulation and the MDP formulation are included.

As seen in the results section, the DQN-based policies failed to learn the desired behaviour despite hundreds of trials to setup the reward function and simulation in a more suitable way. It was observed that the baseline accumulates more rewards compared to the DQN policies during evaluation. This shows that the issue might not be the reward function but rather the algorithm or the training setup. Another aspect regarding the rewards is that they grew to be quite hard to explore during training because they depend on certain observation combinations that only occur in rare cases. On the other hand, simple rewards aren't sufficient for this use case due to having to optimize multiple objectives simultaneously, namely safety and comfort. Nevertheless rewards should be kept as simple as possible and shouldn't need to describe the correct behaviour for each possible situation separately because then we lose the value of using RL completely. Considering that designing the reward function based on intuition even for this simplified model was problematic, motivates the search for alternative methods for setting up the rewards.

One of the key aspects in using deep RL for this task lies in the robustness of the learned policy. This was not investigated during this project but once some noise is added to the observations - as it would happen in a real life setting - the decision-tree baseline would fail miserably because of its hard-coded nature. On the other hand, a fine-tuned DQN based policy would be able to handle such discrepancies in the input - at least in theory. Also, designing a decision tree for a dozen, discrete

observation variables is doable, however, doing this for a larger state space can easily explode in complexity.

Regarding the simulated environment and MDP formulation, there are several assumptions made. Firstly, it is assumed that the observations are fully observable and reliable. This assumption is partly broken when autonomous system failures occur since that triggers the re-computation of TTA4U and TTA4F, rendering the values of those variables in the time steps before the failure inaccurate. Then, it is assumed that such observations can be computed, which is not a straightforward task in reality. For instance, TTDU is one of those variables which are hard to compute.

7 Conclusion and Future work

To conclude, this research introduces a basic MDP environment for optimizing driving entity switching when the autonomous system encounters an error or runs out of its ODD. A baseline decision-tree-based policy is designed and evaluated, showing adequate performance. Different DQN implementations are also trained to find policies for this use case, however their performance lags behind. It appears to be the case that using DQN to handle this multi-objective decision problem using an intuition-based reward function cannot learn a suitable policy.

To achieve comparable results to the baseline, several aspects need more investigation: simplifying the rewards, implement and train algorithms more suitable for multi-objective RL, such as possibly [19], alter the simulation to have more relevant training samples (and less cases for having to do nothing) for easier learning. There are other possibilities for extending this research as well. One direction is looking into when the driver should approve the action of the agent and when it's preferable to take a decision autonomously. Another important aspect is to further investigate which observations can be obtained in reality and make their simulation more realistic. In order to provide more reliability, introducing constraints on the action space could be an option as well. Also, in order for the mediator system to cooperate with the driver, AI explainability will be crucial. Finally, combining the modeling of this use case with others such as encountering uncomfortable events will be also important.

References

[1] MEDIATOR project. *MEDIATOR About the project*. 2021. URL: https://mediatorproject.eu/about/about-mediator (visited on 04/19/2021).

[2] Yang Li and Matthijs Spaan. *Overview of Benchmarks for Reinforcement Learning in Switching Control between Driver and Automated System project*. 2021. URL: https://projectforum.tudelft.nl/course_editions/39/projects/1098 (visited on 04/19/2021).

[3] Franco van Wyk, Anahita Khojandi, and Neda Masoud. "Optimal switching policy between driving entities in semi-autonomous vehicles". In: *Transportation Research Part C: Emerging Technologies* 114 (2020), pp. 517–531.

[4] D Vermunt. "A Markov Decision Process approach to human-autonomous driving control logic". In: (2020). URL: https://repository.tudelft.nl/islandora/object/uuid:ddeffcbb-5dab-4bf1-9c13-af3db7308127.

[5] Volodymyr Mnih et al. "Asynchronous methods for deep reinforcement learning". In: *International conference on machine learning*. PMLR. 2016, pp. 1928–1937.

[6] Alexander Eriksson and Neville A Stanton. "Takeover time in highly automated vehicles: non-critical transitions to and from manual control". In: *Human factors* 59.4 (2017), pp. 689–705.

[7] Thierry Bellet et al. "From semi to fully autonomous vehicles: New emerging risks and ethico-legal challenges for human-machine interactions". In: *Transportation research part F: traffic psychology and behaviour* 63 (2019), pp. 153–164.

[8] Natasha Merat et al. "Transition to manual: Driver behaviour when resuming control from a highly automated vehicle". In: *Transportation research part F: traffic psychology and behaviour* 27 (2014), pp. 274–282.

[9] Natasha Merat et al. "Human factors of highly automated driving: results from the EASY and CityMobil projects". In: *Road vehicle automation*. Springer, 2014, pp. 113–125.

[10] Jin Cui et al. "A review on safety failures, security attacks, and available countermeasures for autonomous vehicles". In: *Ad Hoc Networks* 90 (2019), p. 101823.

[11] Iago Pachêco Gomes and Denis Fernando Wolf. "Health Monitoring System for Autonomous Vehicles using Dynamic Bayesian Networks for Diagnosis and Prognosis". In: *Journal of Intelligent & Robotic Systems* 101.1 (2021), pp. 1–21.

[12] Saeid Safavi et al. "Multi-Sensor Fault Detection, Identification, Isolation and Health Forecasting for Autonomous Vehicles". In: *Sensors* 21.7 (2021), p. 2547.

[13] Eric Thorn et al. *A framework for automated driving system testable cases and scenarios*. Tech. rep. United States. Department of Transportation. National Highway Traffic Safety ..., 2018.

[14] Robert Platt. *Markov Decision Processes*. https://www.ccs.neu.edu/home/rplatt/cs5335_fall2017/slides/mdps.pdf. 2017.

[15] Greg Brockman et al. *OpenAI Gym*. 2016. eprint: arXiv:1606.01540.

[16] Antonin Raffin et al. *Stable Baselines3*. https://github.com/DLR-RM/stable-baselines3. 2019.

[17] Ashley Hill et al. *Stable Baselines*. https://github.com/hill-a/stable-baselines. 2018.

[18] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/.

[19] Thanh Thi Nguyen et al. "A multi-objective deep reinforcement learning framework". In: *Engineering Applications of Artificial Intelligence* 96 (2020), p. 103915. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2020.103915. URL: https://www.sciencedirect.com/science/article/pii/S0952197620302475.

A Hyper-parameter optimization results

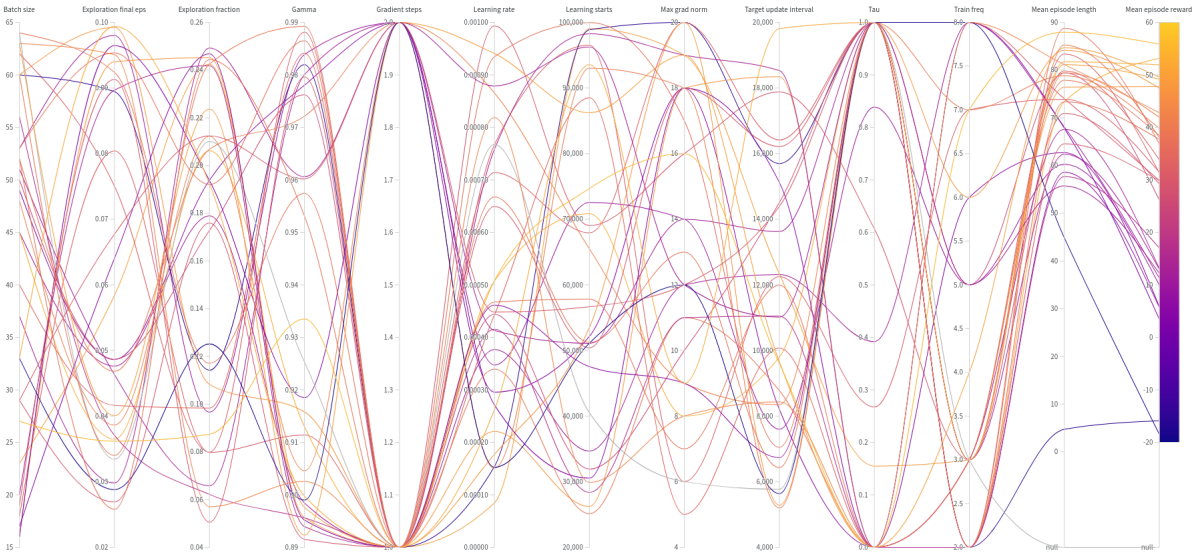


Figure 7: Weights & Biases - Sweep of 33 runs