Autonomous Driving car project

Alessandro Di Frenna

30/09/2025

Abstract

This paper describes the implementation and analysis of different Reinforcement Learning (RL) approaches to solve the `highway-v0` simulated environment of the `gymnasium` package. The performance of a control **Baseline**, the two-way **Deep Q-Network (DQN)** algorithm (standard and with manual feature selection) and **Tabular Q-Learning** are compared. The final section presents a comparison table to evaluate the effectiveness of each model in terms of reward, success rate, and stability.

1 Baseline Policy Description

This section presents a rule-based heuristic baseline policy for autonomous highway driving.

1.1 Design Principles

The heuristic policy follows three fundamental principles:

1. **Safety First**: Avoid collisions through safe distance maintenance
2. **Progress**: Maintain forward velocity when safe to do so
3. **Efficient Lane Usage**: Change lanes only when blocked by slower traffic

1.2 Policy Logic

The policy implements a simple three-stage decision process:

1.2.1 Stage 1: Identify Front Vehicle

The policy first identifies the closest vehicle directly ahead in the current lane:

$$v_{\text{front}} = \arg\min_i \{x_{\text{rel},i} \mid \text{presence}_i = 1, x_{\text{rel},i} > 0, |y_{\text{rel},i}| < \tau_{\text{lane}}\} \tag{1}$$

where $\tau_{\text{lane}} = 0.3$ defines the lateral tolerance for considering a vehicle in the same lane.

1.2.2 Stage 2: Assess Situation

If a front vehicle is detected, compute two critical metrics:

Distance Check:

$$\text{too\_close} = \begin{cases} \text{True} & \text{if } x_{\text{rel}} < d_{\text{safe}} \\ \text{False} & \text{otherwise} \end{cases} \tag{2}$$

where $d_{\text{safe}} = 0.2$ represents the minimum safe following distance.

Relative Velocity Check:

$$\text{approaching} = \begin{cases} \text{True} & \text{if } v_{x,\text{rel}} < -v_{\text{threshold}} \\ \text{False} & \text{otherwise} \end{cases} \tag{3}$$

where $v_{\text{threshold}} = 0.1$ indicates the ego vehicle is significantly faster than the front vehicle.

1.2.3 Stage 3: Action Selection

The policy follows this simple decision tree:
if the lane ahead is clear with no vehicles detected, the policy accelerates to maintain highway speed. If a vehicle is dangerously close (within 0.15 units), emergency braking is triggered immediately. If a vehicle is within the safe distance threshold or approaching too quickly, the policy attempts a lane change to either adjacent lane, prioritizing left over right. When no safe lane change is available, the policy brakes to increase following distance. In all other cases where sufficient clearance exists, the policy accelerates to maintain forward progress.

Algorithm 1 Simple Heuristic Highway Policy

Input: Observation $\mathbf{O} \in \mathbb{R}^{5\times5}$
Output: Action $a \in \mathscr{A}$

$v_{\text{front}} \leftarrow \text{GetFrontVehicle}(\mathbf{O})$

if $v_{\text{front}} = \text{null}$ **then**
  $a = 3$ {No vehicle ahead: accelerate}
end if

$d \leftarrow x_{\text{rel}}(v_{\text{front}})$
$v_{\text{rel}} \leftarrow v_{x,\text{rel}}(v_{\text{front}})$

if $d < d_{\text{critical}}$ **then**
  $a = 4$ {Emergency: brake immediately}
end if

if $d < d_{\text{safe}}$ **or** $v_{\text{rel}} < -v_{\text{threshold}}$ **then**
  $a_{\text{lane}} \leftarrow \text{TryLaneChange}(\mathbf{O})$
  if $a_{\text{lane}} \neq \text{null}$ **then**
    a_{lane} {Safe lane change available}
  else
    $a = 4$ {Blocked: brake}
  end if
else
  $a = 3$ {Safe distance: accelerate}
end if

1.3 Lane Change Safety Check

For each potential target lane, the policy examines all observed vehicles to verify safety. A lane is considered unsafe if any vehicle occupies it (lateral position within 0.3 units of the target lane's center) and is too close longitudinally (within 0.25 units). This safety margin accounts for vehicles beside, slightly ahead, or slightly behind the ego vehicle. The policy only executes a lane change when no vehicles in the target lane violate these conditions, prioritizing safety over opportunity.

Algorithm 2 TryLaneChange

Input: Observation $\mathbf{O}$, current lane ℓ_{current}
Output: Action a or null

for $\ell_{\text{target}} \in \{\ell_{\text{current}} - 1, \ell_{\text{current}} + 1\}$ **do**
  if IsLaneSafe($\mathbf{O}, \ell_{\text{target}}$) **then**
    if $\ell_{\text{target}} = \ell_{\text{current}} + 1$ **then**
      $a = 0$ {Lane left}
    else if $\ell_{\text{target}} = \ell_{\text{current}} - 1$ **then**
      $a = 2$ {Lane right}
    end if
  end if
end fornull {No safe lane available}

Based on the situation assessment, the policy selects an action following a priority hierarchy.

1.4 Implementation Details

The policy was implemented in Python with the following key functions:

- `get_front_vehicle()`: Scans observations for vehicles ahead in current lane
- `is_lane_safe()`: Checks clearance in target lane before lane changes
- `act()`: Main decision function implementing the algorithm above

All distance and velocity comparisons use normalized values from the environment's observation space, ensuring the policy is scale-invariant across different highway configurations.

2 Deep Q-Network (DQN)

The DQN agent is trained to solve the `highway-v0` environment. This section describes the environment configuration, the neural network architecture, and the training procedure.

2.1 Environment Configuration `highway-v0`

The state is defined by the `Kinematics` type, representing the kinematic state of surrounding vehicles:

- `vehicles_count`: 5(number of observed vehicles, including the agent);
- `lanes_count`: 5 (number of highway lanes);
- `features`: `[presence, x, y, vx, vy]` (kinematic and presence features).

Thus, the state dimensionality for the neural network is:

$$S_{dim} = vehicles\_count \times features = 5 \times 5 = 25.$$

2.1.1 Reward Function

The reward function guides the agent's behavior and is defined with the following weights:

- `collision`: -10.0 (penalty for collisions);
- `high_speed`: $+8.0$ (reward for maintaining speed between 22–23 m/s);
- `acceleration`: $+1.5$ (reward for acceleration usage);
- `jerk`: -0.5 (penalty for abrupt acceleration changes);
- `lane_change`: -0.2 (penalty for unnecessary lane changes);
- `off_road_terminal`: `True` (terminates the episode if the vehicle leaves the road).

2.2 Core Components of DQN

2.2.1 Replay Buffer

The replay buffer [1] is implemented as a bounded queue (`deque`, 50,000 transitions). Transitions $(s_t, a_t, r_{t+1}, s_{t+1}, d)$ are sampled in batches of 32 during training.

2.2.2 ε-greedy Exploration Strategy

Exploration is controlled by an exponential decay [2]:

$$\varepsilon(t) = \varepsilon_{end} + (\varepsilon_{start} - \varepsilon_{end}) e^{-t/\tau},$$

with $\varepsilon_{start} = 1.0$, $\varepsilon_{end} = 0.01$, and $\tau = \{25000\}\, steps$.

$\tau = 25000$: The exploration decay exhibits a near-linear profile, ensuring the agent maintains a high degree of exploration throughout most of the training process. Consequently, exploitation of the learned policy becomes the dominant behavior only towards the final stages of training.

2.3 Neural Network Architecture

The action-value function $Q(s,a)$ is approximated by a multilayer perceptron (MLP) [1] with three linear layers and ReLU activations [3]:

1. Input: $25 \to d$ neurons;
2. Hidden: $d \to d$ neurons;
3. Output: $d \to 5$ neurons (number of actions).

Dropout (p=0.2) [4] is applied after each ReLU activation.
d = $\{256, 128, 64, 32\}$

2.4 DQN Hyperparameters Analysis

The selected hyperparameters are optimized for rapid and stable learning in a moderately complex environment, prioritizing fast convergence over absolute maximal performance.

- **TRAIN_FREQ = 4**: Reduces computation and slightly decorrelates consecutive experiences, improving stability [1].
- **BATCH_SIZE = 32**: Balances gradient accuracy and update frequency; smaller batches introduce gradient noise but help escape local minima [5].
- **TARGET_UPDATE = 1000**: Frequent target network updates prevent stale Q-value targets, accelerating early-stage convergence [1].

Learning Control

- $\gamma = 0.95$: Slightly short-sighted, emphasizing immediate rewards and reducing target variance for more stable training [2].

- **LR = 5e-4**: Moderately high learning rate for fast convergence; stability maintained by frequent target updates and smaller γ [6].

3 DQN with Manual Feature Selection

The parameters remain equal to the previous DQN version since the difference of input dimension is quite trivial, therefore we can adopt the same training parameters for the same reasons.

3.1 Feature Extraction and Motivation

The raw observations of the environment (`obs_raw`) typically include the full kinematic description of all vehicles in the scene. While this representation is complete, it is also highly redundant and computationally inefficient. Therefore, a dedicated feature extraction pipeline has been implemented to emphasize only the most relevant spatial and temporal information [7].

3.2 Implementation Details

The function first filters out vehicles that are not present on the drivable lanes, retaining only those that can influence the ego-vehicle's behavior. The ego-vehicle is always considered as the reference point and its lane index is extracted as the primary categorical feature.

Next, the relative positions of surrounding vehicles are computed. For each vehicle, the algorithm evaluates its longitudinal and lateral offsets with respect to the ego-vehicle. The four most safety-critical directions — front, rear, left, and right — are considered. For each of these directions, the function stores the minimum observed distance and the corresponding relative coordinate. This ensures that only the closest threats or constraints are retained in the final state vector, while distant vehicles are ignored.

To enrich the purely geometric representation with temporal information, the feature vector also includes the *Time-to-Collision (TTC)* [8] with the nearest front and rear vehicles in the same lane. This measure accounts for both relative velocities and accelerations, capturing the dynamics of potential collisions rather than relying exclusively on instantaneous distances.

3.3 Motivation and Rationale

The rationale behind this feature extraction strategy is for:

1. **Focus on surrounding vehicles.** By explicitly representing the nearest vehicles in the front, rear, left, and right directions, the agent is provided with the minimal yet sufficient information to evaluate lane-change feasibility, overtaking opportunities, and collision risks [9].

2. **Integration of temporal safety metrics.** Purely spatial distances cannot fully capture collision risk. The inclusion of TTC values with the leading and following vehicles in the same lane introduces a predictive component, enabling the agent to anticipate and avoid dangerous situations before they materialize [8].

3.4 Final State Representation

The final feature vector is an 11-dimensional array that integrates lane information, minimum distances in all four directions, relative coordinates of the nearest vehicles, and TTC values, which will feed the different NNs of the DQN.

4 Q-Learning

The training parameters like gamma and epsilon-decay are the same of the previous models. The learning rate alfa is set to 0.1 [2].

4.1 State Discretization via Binning

Once the raw features are extracted, they must be encoded into a finite set of states in order to make the environment compatible with tabular reinforcement learning methods [10]. For this purpose, each continuous feature (e.g., distances, lateral offsets, time-to-collision) is discretized into non-uniform intervals (bins). The mapping is performed by the `map2States` wrapper, which associates each feature vector to a unique discrete index.

4.2 Design of Bin Intervals

The bin thresholds are generated according to an exponential law. The motivation for this design lies in the observation that the criticality of a traffic scenario increases significantly as distances and time-to-collision values approach zero [9]. In such cases, small variations in the measurements correspond to large changes in risk and, consequently, in the optimal decision. Conversely, when distances or collision times are large, the system dynamics evolve more smoothly and coarse-grained state distinctions are sufficient.

Formally, let $f(x) = e^{-kx}$ be the exponential decay function, with k controlling the rate of decay. For each feature, thresholds are obtained by sampling $f(x)$ on a fixed interval and scaling the values to the appropriate range (positive and negative). This ensures that bins are *densely concentrated* near zero and become progressively *wider* as the distance or time horizon increases.

4.3 Interpretation

This discretization scheme has two key advantages:

1. **High resolution in critical regions.** When the ego-vehicle is very close to another vehicle (in distance or in time-to-collision), even minor differences in relative position or velocity may require drastically different actions (e.g., braking vs. lane change). By allocating a large number of bins in this region, the agent acquires higher decision accuracy and finer control over safety-critical maneuvers [9].

2. **Reduced complexity in safe regions.** At larger distances or longer collision times, the precise value of the feature becomes less relevant, since the immediate risk is negligible. Here, fewer states are sufficient to represent the environment, avoiding unnecessary state-space explosion [2].

4.4 Comparative Analysis of DQN Architectures for Highway Driving

4.4.1 Network Architecture Specifications

Four distinct Deep Q-Network (DQN) architectures were evaluated on the highway driving task, differing in both network capacity and input representation. All networks employ fully-connected layers with ReLU activations [3] and are trained using

standard DQN with experience replay [1]. The architectures are specified as follows:

Input Representations:

- **Flattened Input (Net-1, Net-2):** Direct vectorization of the (5×5) observation matrix, yielding a 25-dimensional input vector containing all raw features: $[\text{presence}, x_{\text{rel}}, y_{\text{rel}}, v_{x,\text{rel}}, v_{y,\text{rel}}]$ for each of the 5 vehicles. 2 Hidden layers each (256-256,) (128-128), respectively.
- **Engineered Features (Net-3, Net-4):** Hand-crafted 11-dimensional feature vector including: distances to nearest vehicles in each lane, relative velocities, lane occupancy indicators, and time-to-collision metrics. This representation reduces dimensionality while incorporating domain knowledge [7]. 2 Hidden layers each (64-64,) (32-32), respectively.

5 Training Performance Analysis

5.1 Net-1: 256-256 with Raw Input

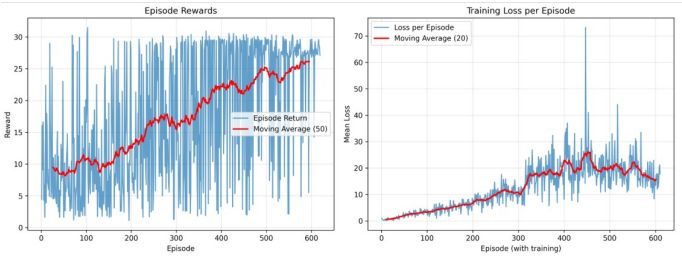

Figure 1: **DQN Standard** – Fully Connected (**FC**(**256˘256**)), raw input

5.1.1 Episode Rewards

The largest network demonstrates a clear upward trend in episode rewards, reaching a moving average of approximately 27-28 by episode 600. The learning curve exhibits:

- **Steady convergence**: Monotonic improvement with minimal plateaus
- **Late-stage stability**: Convergence appears to occur around episode 500-600

This behavior is characteristic of sufficient network capacity successfully learning the value function. The high parameter count (72k) provides adequate representational power for the relatively low-dimensional state space.

5.1.2 Training Loss Analysis

The loss gradually increases from $\sim$2 to $\sim$20 over 600 episodes before stabilizing.

The stabilization around episode 600 coincides with reward convergence, suggesting the network has learned to approximate the true action-value function $Q^*(s,a)$.

This is *healthy learning behavior*. The rising loss reflects the agent discovering that achievable returns are higher than initially estimated. Once these corrected Q-values stabilize, the loss plateaus.

5.2 Net-2: 128-128 with Raw Input

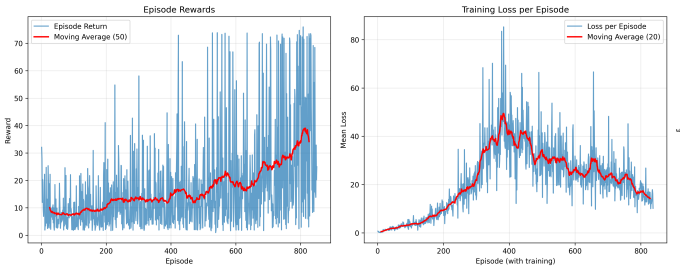

Figure 2: **DQN Standard** – Fully Connected (**FC**(**128˘128**)), raw input

5.2.1 Episode Rewards

The medium-large network achieves the highest final performance with rewards reaching 35-40. Key observations:

This architecture appears to strike an optimal balance between capacity and regularization for this task. The reduced parameter count (20k vs 72k) may provide better generalization by preventing overfitting to specific traffic scenarios encountered during training.

5.2.2 Training Loss Analysis

Net-2 exhibits a dramatic loss increase to 40-50 during episodes 300-500, followed by gradual decline to $\sim$15. Same considerations of the aforementioned case.

5.3 Net-3: 64-64 with Feature Engineering

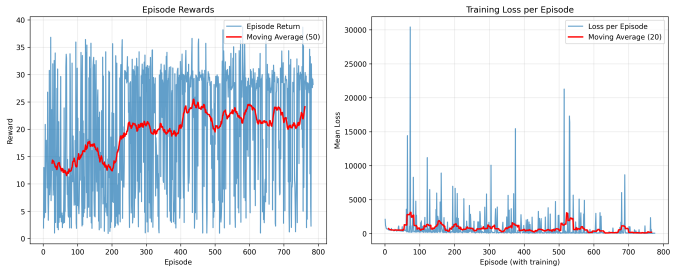

Figure 3: **DQN** – Fully Connected (**FC**(**64˘64**)), feature selection (**11** feats)

5.3.1 Episode Rewards

This network demonstrates robust learning with final rewards around 22-25.

The engineered features clearly provide valuable inductive bias, enabling a smaller network (5k parameters) to learn effectively. However, the performance ceiling suggests that feature engineering may have discarded information relevant for optimal policy learning.

5.3.2 Training Loss Analysis

The loss remains consistently low (<1000) with occasional spikes. This stability arises from Effective feature representation which reduce input noise and irrelevant variance,and appropriate capacity: The 64-64 architecture is well-matched to the 11-dimensional input. While low loss appears desirable, it may indicate the network is learning a simpler but suboptimal policy that avoids challenging states, resulting in lower final rewards compared to Net-2.

5.4 Net-4: 32-32 with Feature Engineering

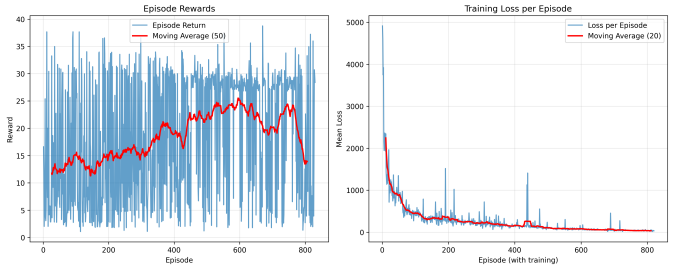

Figure 4: **DQN** – Fully Connected (**FC**(**32˘32**)), feature selection (**11** feats)

5.4.1 Episode Rewards

The smallest network shows concerning training dynamics: The reward curve exhibits high variance with a slowly increasing moving average that eventually plateaus and declines. This pattern indicates that the agent initially benefits from exploration but later converges toward a suboptimal policy. The degradation in performance suggests that the learned Q-function fails to generalize effectively, likely due to limited model capacity (2k parameters) and instability arising from the interaction of function approximation, bootstrapping, and off-policy learning—an implicit manifestation of the deadly triad.

5.4.2 Training Loss Analysis

Net-4 demonstrates severely probelms during training:

1. **Initial decline**: Loss drops rapidly from 2500 to 200 (episodes 0-200)
2. **Divergence phase**: Sudden spikes to 1500+ occur after episode 400
3. **Persistent instability**: Never achieves stable convergence

The mean training loss decreases rapidly and stabilizes at low values, reflecting numerical convergence to the Bellman targets rather than true policy improvement. This apparent stability can be misleading: the network minimizes prediction error on biased targets generated through bootstrapping and off-policy sampling, rather than capturing accurate action-value estimates. As a result, the loss appears well-behaved even as policy performance deteriorates.

6 Comparative Analysis

6.1 Capacity-Performance Relationship

Table 1: Final Performance Metrics (Episode 800)

Architecture	Parameters	Avg Reward	Final Loss
Net-1 (256-256)	72k	27-28	20
Net-2 (128-128)	20k	35-40	15
Net-3 (64-64)	5k	22-25	<1000
Net-4 (32-32)	2k	15-20	Unstable

Why Loss Increases During Successful Learning: As the policy improves, the agent: visits higher-value states with larger expected returns so updates Q-values upward to reflect these discoveries

During this adjustment period, the online network's Q-values ($Q(s,a;\theta)$) lag behind the increasing targets (y), causing:

Why Loss Eventually Decreases: Once the policy stabilizes, the distribution of experienced states becomes stationary, and the Bellman operator becomes contractive. The Q-network converges to a fixed point where $Q(s,a) \approx r + \gamma \max_{a'} Q(s',a')$, minimizing TD error.

6.2 Comparative Results Table

Table 2: Comparative summary of performance results.

Model	**Avg. Episode Length** ($\pm\sigma$)	**Avg.Reward** ($\pm\sigma$)	**Success R.(%)**	**Reward(95% C.I.)**
Baseline – FC(256–256)	37.02 ± 8.63	26.07 ± 6.26	87.5	24.34 – 27.81
DQN FC(128–128)	40.00 ± 0.00	27.87 ± 2.31	95.0	27.23 – 28.51
DQN FC(256–256)_2	40.00 ± 0.00	26.82 ± 0.97	100.0	27.67 – 28.20
DQN FC(64–64), 11 feats	40.00 ± 0.00	27.93 ± 0.95	100.0	27.67 – 28.20
DQN FC(32–32), 11 feats	34.56 ± 9.72	24.65 ± 0.56	85.0	22.50 – 26.81
Q-Learning	18.22 ± 12.23	15.81 ± 15.2	67.2	9.86 – 22.89

To compare models, the following metrics were employed:

- Average episode length ($\pm\sigma$);
- Average reward per episode ($\pm\sigma$);
- Success rate (% of episodes without collisions or road exits);
- Average reward with 95% confidence interval.

This results are obtained running the environment with:

- Environment duration (steps): 40
- Number of lanes (`lanes_count`): 3
- Number of vehicles (`vehicles_count`): 5 The environment was sampled for n = 50 episodes

7 Conclusions

Based on the reported results, there is no statistically significant evidence to conclude that any single agent configuration is definitively superior to the others. The core reason for this scientific stance is the proximity of the mean reward values across all tested models.

Despite this, the DQN–FC(64–64) agent with feature selection (11 features) achieves the highest mean reward of 27.93, showing only a slight advantage.

Across all training scenarios, the agents consistently learned a fundamental safety policy: decelerating when approaching a lead vehicle to avoid a collision. However, they failed to develop more proactive, high-performance strategies, such as overtaking to maximize speed.

The agents consistently defaulted to a conservative "brake-and-wait" behavior. This suggests a convergence to a safe but suboptimal local optimum. It is plausible that this simple, defensive strategy is the easiest to learn, as the immediate and severe penalty for crashing heavily outweighs the more complex, delayed reward of a successful overtake.

To encourage a more dynamic driving policy, future work should focus on recalibrating the reward function. By increasing the weights for maintaining high speed and forward progress, the agent could be incentivized to explore and master more effective overtaking maneuvers.

References

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

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press, 2018.

[3] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2011, pp. 315–323.

[4] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[5] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," in *Proc. 5th Int. Conf. Learning Representations (ICLR)*, 2017.

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learning Representations (ICLR)*, 2015.

[7] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-Baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[8] K. Vogel, "A comparison of headway and time to collision as safety indicators," *Accident Analysis & Prevention*, vol. 35, no. 3, pp. 427–433, 2003.

[9] M. Althoff, O. Stursberg, and M. Buss, "Safety assessment of driving behavior in multi-lane traffic for autonomous vehicles," in *Proc. IEEE Intelligent Vehicles Symposium*, 2009, pp. 893–900.

[10] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.