Autonomous Highway Driving with Deep Reinforcement Learning

Antonio Tangaro [1]

Abstract

This project addresses autonomous highway driving using Deep Reinforcement Learning. Five algorithms are implemented and compared: DQN, Double DQN, Dueling DQN, D3QN, and PPO. A Time-To-Collision (TTC) heuristic baseline is developed as comparison. Results show that all RL agents significantly outperform the baseline, achieving approximately +15 points in mean return and reducing crash rates from 89% to 0–2%. Dueling DQN achieves the best performance with 28.30 mean return and 0% crash rate. Bonus experiments investigate reward shaping for smoother driving behavior, revealing that naive modifications can degrade performance and that correct observation threshold calibration is critical. An ActionSmoother post-processing mechanism proves more effective than reward modification.

1. Introduction

This project tackles autonomous highway driving using the `highway-env` simulation environment (Leurent, 2018). The agent must navigate a multi-lane highway populated with vehicles following the Intelligent Driver Model (IDM), maximizing speed while avoiding collisions. The observation consists of a kinematic matrix representing the ego vehicle and the 5 nearest vehicles, with each row containing: presence flag, longitudinal position x, lateral position y, longitudinal velocity v_x, and lateral velocity v_y. Features are normalized according to the environment's observation configuration (bounded scaling). The action space is discrete with 5 actions: change lane left, idle, change lane right, accelerate, and decelerate.

The project implements five DRL algorithms spanning both value-based and policy gradient methods: DQN (Mnih et al., 2015), Double DQN (Van Hasselt et al., 2016), Dueling DQN (Wang et al., 2016), D3QN (combining Double, Dueling, and Prioritized Experience Replay (Schaul et al., 2016)), and PPO (Schulman et al., 2017). This selection allows analysis of off-policy versus on-policy trade-offs and the impact of architectural improvements within the DQN family.

[1]University of Padova, Italy. Correspondence to: Antonio Tangaro <antonio.tangaro@studenti.unipd.it>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

2. Critical Issues

The highway driving task presents several challenges that must be addressed for successful learning.

Partial Observability. The agent perceives only the 5 nearest vehicles within its kinematic representation. Vehicles beyond this range remain invisible, preventing anticipation of traffic patterns that may become relevant. This limited field of view requires the agent to develop robust decision-making under uncertainty rather than relying on complete environmental knowledge.

Sparse Reward Signal. The reward structure creates credit assignment difficulties. Collision penalties (-1) occur only at episode termination, while continuous rewards for speed ($\sim$0.4 scaled by velocity) and right-lane preference ($\sim$0.1) are comparatively small. The agent must learn to associate early decisions with eventual crashes that may occur many timesteps later.

Action Jittering. When Q-values for different actions are numerically similar—a frequent occurrence on open highway sections—small estimation errors cause rapid oscillation between lane change actions. This results in unrealistic zigzag behavior where the agent switches lanes multiple times per second without apparent reason. This issue motivated the development of an ActionSmoother mechanism for evaluation.

Non-Stationary Dynamics. Other vehicles follow the IDM with stochastic components, creating a dynamic environment where the optimal response depends on continuously changing traffic configurations. The agent cannot memorize fixed patterns but must generalize across diverse scenarios.

3. Baseline Policy

A meaningful baseline must represent an "educated guess"—a reasonable approach that a domain expert might implement without machine learning. The Time-To-Collision (TTC) metric satisfies this criterion as it forms the foundation of modern Advanced Driver Assistance Systems (ADAS) used in production vehicles for collision warning and automatic emergency braking (Minderhoud & Bovy, 2001).

3.1. TTC Computation

For each vehicle i in lane l ahead of the ego vehicle, TTC is computed as:

$$\mathrm{TTC}_{i,l} = \frac{x_i}{v_{ego} - v_i}, \quad \text{if } v_{ego} > v_i \tag{1}$$

where x_i is longitudinal distance and v denotes velocity. When $v_{ego} \leq v_i$, the vehicle is not approaching and TTC is set to infinity. The minimum TTC across all vehicles in each lane determines the "safety" of that lane.

3.2. Heuristic Policy

The TTC heuristic operates according to the following procedure:

1. Compute TTC_l for each lane $l \in \{\text{left}, \text{current}, \text{right}\}$
2. If current lane TTC $< \tau_{crit} = 2.0s$:
   - Find lane l^* with maximum TTC (ties broken by preferring right lane)
   - If $l^* \neq$ current and $\mathrm{TTC}_{l^*} > \mathrm{TTC}_{\text{current}}$: change to l^*
   - Else: decelerate (no safe lane available)
3. Else if current lane TTC $< \tau_{warn} = 4.0s$: decelerate
4. Else: accelerate toward maximum speed

Lane changes are only executed if the target lane exists (not at road boundary). The policy does not check for vehicles approaching from behind, which contributes to crashes when merging into occupied lanes.

3.3. Why TTC is Reasonable

The TTC baseline is reasonable because: (1) it uses a physics-based safety metric employed in real vehicles; (2) it prioritizes collision avoidance—the primary objective; (3) it attempts to maintain speed when safe; and (4) it makes decisions based on observable quantities without requiring learning. However, TTC is purely *reactive*: it responds only to imminent danger and cannot anticipate developing situations.

3.4. Baseline Results

Table 1 presents baseline evaluation over 100 episodes. The TTC heuristic achieves 13.28 mean return but suffers from 89% crash rate. This high crash rate occurs because reactive decisions often lead to situations where all escape routes become simultaneously blocked. Random policy serves as lower bound with 8.57 return and 96% crash rate.

Table 1. Baseline evaluation (100 episodes, seed $e-1$ for episode e)

POLICY	RETURN	STD	CRASH
TTC HEURISTIC	13.28	8.87	89%
RANDOM	8.57	6.66	96%

4. Reinforcement Learning Algorithms

4.1. DQN

Deep Q-Network (Mnih et al., 2015) approximates the action-value function $Q(s, a; \theta)$ using a neural network. Two key innovations enable stable learning: (1) *experience replay* stores transitions (s, a, r, s') in a buffer and samples mini-batches uniformly, breaking temporal correlations; (2) a *target network* θ^- updated periodically provides stable TD targets:

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

4.2. Double DQN

Standard DQN suffers from overestimation bias because the same network both selects and evaluates actions. Double DQN (Van Hasselt et al., 2016) decouples these operations: the online network selects actions while the target network evaluates them:

$$y = r + \gamma Q(s', \arg\max_{a'} Q(s', a'; \theta); \theta^-) \tag{3}$$

4.3. Dueling DQN

While not covered in course lectures, Dueling DQN (Wang et al., 2016) represents a natural architectural extension addressing a key limitation: in many states, the action choice has minimal impact on outcomes. The architecture decomposes the Q-function into state-value $V(s)$ and advantage $A(s, a)$ streams:

$$Q(s, a) = V(s) + \left(A(s, a) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s, a')\right) \tag{4}$$

The advantage mean subtraction ensures identifiability. This decomposition allows the network to learn which states are inherently valuable regardless of action, improving learning efficiency precisely in highway scenarios where cruising on open road provides similar outcomes across actions.

4.4. D3QN

D3QN combines three complementary improvements: Double Q-learning (reducing overestimation), Dueling architecture (value-advantage decomposition), and Prioritized Experience Replay (Schaul et al., 2016). PER samples transitions proportionally to TD-error magnitude $|\delta|^\alpha$, focusing learning on surprising experiences. Importance sampling weights $w_i = (N \cdot P(i))^{-\beta}$ correct the resulting bias.

4.5. PPO

Proximal Policy Optimization (Schulman et al., 2017) is included for on-policy comparison. It optimizes a clipped surrogate objective preventing destructively large policy updates:

$$\mathcal{L}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 \pm \epsilon)\hat{A}_t\right)\right] \quad (5)$$

where $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{old}}(a_t|s_t)$ is the probability ratio and $\hat{A}_t$ is the generalized advantage estimate.

4.6. Implementation Details

All algorithms use a two-layer MLP with 256 hidden units and ReLU activations. The network architecture for Dueling DQN splits after the first hidden layer into separate value and advantage streams, each with one hidden layer, before combining via the aggregation equation. Key hyperparameters: learning rate 5×10^{-4}, discount $\gamma = 0.99$, batch size 64, replay buffer size 100,000. DQN variants use ϵ-greedy exploration decaying from 1.0 to 0.05 over 10,000 steps. PPO uses 2048-step rollouts with 10 optimization epochs. Random seed fixed to 0 for reproducibility.

5. Experimental Results

5.1. Training Configuration

All algorithms were trained for 50,000 environment transitions (steps) on `highway-v0` with 3 lanes and approximately 50 vehicles. Each step corresponds to one `env.step(action)` call. Evaluation occurs every 5,000 steps over 5 episodes. The simulation runs at accelerated speed (15 Hz policy, no rendering during training).

Training efficiency was improved through CUDA-optimized vectorized environments. Using the `--num_envs` flag on NVIDIA GPUs reduces wall-clock training time by approximately 66% (from ∼1.5h on Apple Silicon/MPS to ∼30min on CUDA) for the same 50,000-step budget. Apple Silicon devices fall back to single-environment training due to MPS multiprocessing limitations.

Table 2. Algorithm comparison (100 episodes with ActionSmoother)

ALGORITHM	RETURN	STD	CRASH
DUELING DQN	**28.30**	0.96	**0%**
D3QN	28.24	0.95	0%
DQN	28.20	0.85	0%
DOUBLE DQN	28.19	0.87	0%
PPO	28.07	1.65	2%
TTC BASELINE	13.28	8.87	89%

5.2. Evaluation Protocol

All reported metrics are computed over 100 evaluation episodes. For reproducibility, episodes use deterministic seeding: episode $e \in \{1, \dots, 100\}$ uses seed $e - 1$, ensuring identical traffic configurations across experiments. The *episodic return* $G = \sum_{t=0}^{T} r_t$ is the undiscounted sum of rewards per episode. *Crash rate* is the fraction of episodes where `info["crashed"]=True` at termination; timeouts are counted as non-crash episodes.

5.3. Algorithm Comparison

Table 2 presents evaluation results over 100 episodes with ActionSmoother enabled (discussed below). All RL algorithms dramatically outperform the baseline, achieving approximately +15 points in return and reducing crash rates from 89% to 0–2%.

Figure 1 shows learning curves. Off-policy methods (DQN family) converge within 5,000–10,000 steps, demonstrating superior sample efficiency. The replay buffer enables learning from each experience multiple times. PPO, being on-policy, requires more environment interactions and shows a gradual improvement curve.

Dueling DQN and D3QN exhibit the lowest variance (std $<$ 1.0), suggesting the value-advantage decomposition provides more stable Q-estimates. PPO shows higher variance (std 1.65) and occasional crashes (2%), consistent with on-policy methods' sensitivity to environmental stochasticity. All algorithms reach similar final performance (∼28 return), indicating the task has a performance ceiling determined by environment dynamics rather than algorithm choice.

5.4. ActionSmoother

During evaluation, all trained agents exhibited "jittery" behavior—rapid oscillation between lane changes when Q-values are similar. This occurs because small numerical differences in Q-values (e.g., $Q(s, \text{left}) = 5.001$ vs $Q(s, \text{idle}) = 5.000$) deterministically select different actions frame-by-frame due to noise in value estimation.

The ActionSmoother addresses this by enforcing a mini-

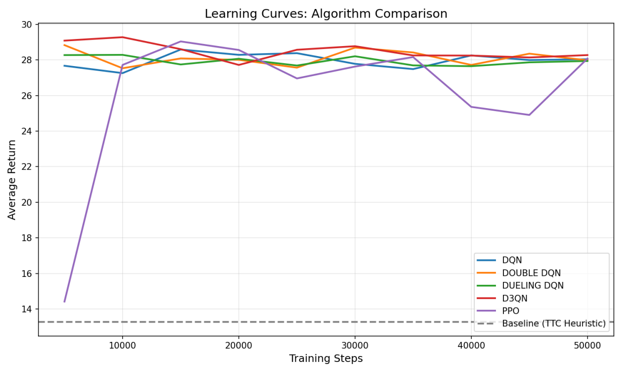

Figure 1. Learning curves. DQN variants converge rapidly while PPO shows gradual improvement with higher variance. Dashed line indicates TTC baseline.

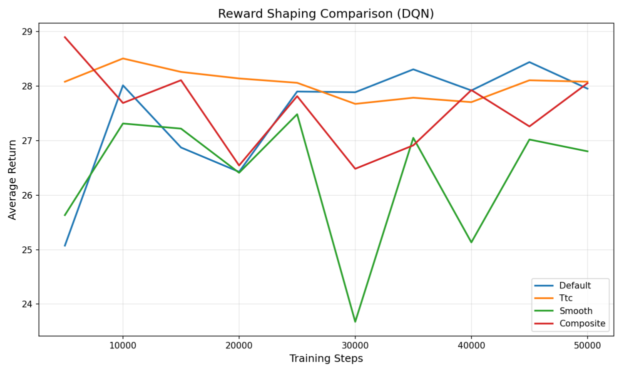

Figure 2. Reward shaping comparison. Default and TTC variants perform similarly; Smooth variant shows degraded performance.

mum cooldown period (default: 3 steps) between consecutive lane changes. It also prevents immediate reversal (LEFT→RIGHT or vice versa) which never represents rational driving behavior. Critically, emergency detection allows bypassing the cooldown when a vehicle is detected very close ahead (normalized longitudinal distance $x < 0.1$, corresponding to the closest decile of the observation range), preserving collision avoidance capability.

Without ActionSmoother, agents achieve similar numerical returns but exhibit unrealistic behavior unsuitable for real-world deployment. The smoother acts as a post-processing filter that removes artifacts of discrete Q-value argmax selection without modifying the underlying learned policy.

6. Bonus Experiments

6.1. Reward Shaping

The jittering problem motivated investigation into reward shaping to encourage smoother driving behavior directly during training.

Naive Approach. Initially, all lane changes were penalized with a fixed penalty (-0.1). This degraded performance because necessary overtaking maneuvers were discouraged, causing the agent to remain stuck behind slow vehicles.

Intelligent Approach. A context-aware reward wrapper was developed that applies: unnecessary lane change penalty (-0.2) only when no slow vehicle is detected ahead; right-lane bonus $(+0.1)$ for staying in rightmost lane when safe; and zigzag penalty (-0.3) for LEFT→RIGHT or RIGHT→LEFT sequences.

A critical challenge emerged in threshold calibration. The normalized observation encodes lateral position as $y \in [0, 1]$ where $y = 0.75$ indicates the right lane (for 3 lanes). Initial experiments used incorrect thresholds ($y > 0.05$ for "right lane") based on incorrect assumptions about the coordinate system. After debugging by printing raw observations, thresholds were corrected to $y > 0.6$ for right lane detection.

Figure 2 shows results. Even with corrected thresholds, reward shaping did not significantly improve learned behavior—the "Smooth" variant shows degraded and unstable performance. The ActionSmoother post-processing proved more effective than reward modification, suggesting the default `highway-env` reward is already well-tuned.

Lesson Learned. Reward shaping requires careful calibration of observation thresholds. Debugging by examining raw observation values is essential before implementing reward modifications. Post-processing (ActionSmoother) can be more robust than reward engineering for behavior correction.

Table 3. Environment variations (DQN, 50k steps)

CONFIG	LANES	VEHICLES	RETURN
EASY	4	20	28.34
DEFAULT	3	50	27.95
DENSE	3	100	27.73
HARD	2	80	24.71

6.2. Environment Variations

Table 3 presents evaluation across different environment configurations using DQN. The "Hard" setting (2 lanes, 80 vehicles) presents significantly greater difficulty due to reduced maneuvering space and higher traffic density. This configuration could benefit from extended training or curriculum learning approaches.

7. Discussion and Conclusions

7.1. Why RL Outperforms the Baseline

The fundamental difference between the TTC baseline and learned policies is *reactive* versus *anticipatory* behavior. The TTC heuristic responds only when danger is imminent, often too late to find safe escape routes. Learned policies develop proactive strategies observable in their behavior: maintaining larger following distances, preferring lanes with better traffic flow, and avoiding configurations where multiple escape routes could become simultaneously blocked.

The +15 point improvement in return and a reduction of 89 percentage points in crash rate (from 89% to 0–2%) demonstrate that deep RL can discover non-obvious driving strategies that substantially outperform hand-crafted heuristics, even those based on established safety metrics.

7.2. Algorithm Selection

For this discrete-action task with moderate state dimensionality (25 features), off-policy value-based methods offer the best trade-off between sample efficiency and final performance. Dueling DQN achieved the best results, validating the architectural insight that separating state-value from action-advantages improves learning when many actions have similar effects.

PPO, while achieving comparable final performance, required more samples and exhibited higher variance. For tasks where sample efficiency is critical (expensive simulators, real-world deployment), the DQN family is preferred. PPO's advantages (stability, ease of hyperparameter tuning) become more relevant in continuous action spaces not explored in this project.

7.3. Lessons Learned

Three key insights emerged from this project: (1) Dueling DQN and D3QN, while not covered in lectures, represent natural extensions of the DQN family that improve stability through value-advantage decomposition; (2) Naive reward shaping can degrade performance—understanding the observation space and threshold calibration is critical; (3) Post-processing (ActionSmoother) proved more robust than reward modification for achieving smooth driving behavior.

The trained agents achieve 28.30 mean return versus 13.28 for the TTC baseline, with crash rate reduced from 89% to 0–2% depending on the algorithm (PPO at 2%).

7.4. Future Work

Promising directions include: continuous action spaces using DDPG (Lillicrap et al., 2016) or TD3 (Fujimoto et al., 2018) for steering and throttle control; multi-agent scenarios modeling interactions between learning agents; and interpretability analysis examining learned Q-values to understand decision-making patterns.

References

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1587–1596. PMLR, 2018.

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

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *Proceedings of the 4th International Conference on Learning Representations*, 2016.

Minderhoud, M. M. and Bovy, P. H. Extended time-to-collision measures for road traffic safety assessment. *Accident Analysis & Prevention*, 33(1):89–97, 2001.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. In *Proceedings of the 4th International Conference on Learning Representations*, 2016.

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

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double Q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, pp. 2094–2100, 2016.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pp. 1995–2003. PMLR, 2016.