Reinforcement Learning for Autonomous Driving in Highway Environment

Kelvin Asu Ekuri, David Daniel, Danil Afonchikov, Ivan Kiselyov

Abstract

This project investigates the application of model-free reinforcement learning (RL) algorithms to autonomous highway driving in a simulated environment. We implement and evaluate four popular RL algorithms: Deep Q-Network (DQN), Proximal Policy Optimization (PPO), Twin Delayed Deep Deterministic Policy Gradient (TD3), and Soft Actor-Critic (SAC). These algorithms were chosen to span both discrete and continuous action spaces, which allows for a deeper understanding of their strengths and limitations when applied to complex driving tasks. Evaluation was conducted using metrics such as average reward, collision rate, and episode length. Our results show that PPO offered the most consistent and safe driving behavior, while SAC demonstrated strong learning capability with minimal safety violations. TD3 achieved zero collisions but at the cost of realistic driving—its policy often drifted off-road. DQN struggled to maintain stability and demonstrated highly erratic performance. This report provides a comprehensive analysis of the algorithmic foundations, their behavior in simulation, and their relative suitability for safe and intelligent autonomous driving.

Key Words: Reinforcement Learning, Autonomous Driving, Deep RL, highway-env, Model-Free Algorithms, SAC, DQN, TD3, PPO

I. Introduction

Autonomous driving is a complex, sequential decision-making problem that involves navigating dynamic environments, interacting with other agents, and optimizing for safety, speed, and efficiency. Reinforcement learning (RL), particularly model-free methods, offers a powerful framework for learning optimal policies through direct interaction with an environment. Unlike supervised learning, RL requires no labeled data. Instead, agents learn by receiving rewards or penalties based on their actions.

In this project, we focus on model-free algorithms due to their simplicity and directness. Model-free RL does not rely on constructing an explicit model of the environment's dynamics, which can be both difficult and computationally expensive in real-world driving. Instead, agents learn optimal behaviors purely from experience. Our choice of algorithms spans both value-based and policy-based methods, as well as discrete and continuous action space algorithms. This duality allows us to explore different paradigms of control and evaluate their impact on driving performance.

II. Approach

The primary objective of this project is to train reinforcement learning agents to learn optimal driving policies for a highway navigation task. We define an optimal policy as one that maximizes long-term reward by balancing speed, safety, and road rules such as lane discipline.

Our approach follows the standard reinforcement learning pipeline. The agent observes the current state of the environment — such as its position, speed, and the locations of nearby vehicles — and selects an action according to its policy. The action influences the environment, resulting in a new state and a scalar reward signal, which the agent uses to improve its policy over time.

Formally, we aim to learn a policy $\pi(a_t|s_t)$ that maps each state s_t to an action a_t, where the goal is to maximize the expected cumulative discounted reward:

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$$

where γ is the discount factor and r_t is the reward at time step t.

We experiment with both discrete and continuous action space algorithms and evaluate how well they learn the policy given the highway environment. In particular, we assess whether the policy learned by each agent results in realistic driving behavior, such as lane following, overtaking, and collision avoidance.

III. System Specifications

Training was conducted on a high-performance workstation configured as follows:

- **GPU:** NVIDIA GeForce RTX 4060 (8GB VRAM)
- **CUDA Version:** 12.7
- **Driver Version:** 566.03

- **CPU:** Intel Core i7
- **RAM:** 32 GB
- **Operating System:** Windows 11

Each RL algorithm was trained for 100,000 timesteps. Total training time across all agents was approximately 28 hours. Among the algorithms, PPO consumed the highest amount of training time due to its use of full episode rollouts and multiple policy update cycles per iteration.

IV. Methods

A. Environment Setup

We utilized the `highway-fast-v0` environment from the `highway-env` package, which simulates fast-paced highway driving involving lane changes, interactions with other vehicles, and the requirement to maintain high speed while avoiding collisions.

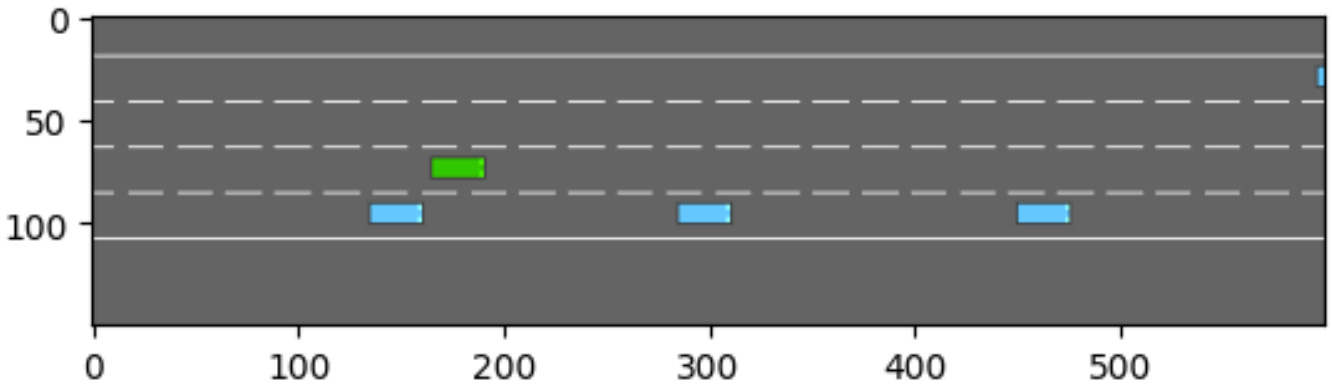

Fig. 1: Highway Environment

1) Environment Configuration

```
env_config = {
    "screen_width": 608,
    "screen_height": 160,
    "normalize_reward": True,
    "lanes_count": 4,
    "vehicles_count": 30,
    "duration": 40,
    "reward_speed_range": [20, 30],
    "collision_reward": -2.0,
    "right_lane_reward": 0.1,
    "high_speed_reward": 0.4,
    "lane_change_reward": 0,
}
```

This environment configuration was carefully tuned to simulate realistic and challenging highway driving conditions:

- **screen_width** and **screen_height**: These define the pixel resolution of the observation space when visual rendering is enabled. A resolution of 608x160 provides a wide horizontal field of view necessary for highway vision while keeping computational load minimal.
- **normalize_reward**: Set to `True` to ensure that reward values are scaled and comparable across agents, which aids in training stability and convergence.
- **lanes_count**: We chose 4 lanes to represent a moderately dense highway, providing enough space for lane changes while still presenting traffic constraints.
- **vehicles_count**: With 30 vehicles, the environment is sufficiently populated to require meaningful interactions and decision-making. This level creates the need for frequent overtakes and risk assessment.
- **duration**: Each episode lasts for 40 time steps, allowing agents to make several tactical and strategic decisions. This duration balances computational efficiency with sufficient learning signal per episode.
- **reward_speed_range**: The agent receives positive rewards when its speed falls between 20 and 30 units. This range encourages high-speed driving that is safe and within control. It motivates the agent to avoid excessive slowing down.

- **collision_reward**: A penalty of -2.0 is imposed for each collision. This negative reward is strong enough to discourage crashes but not so large that it completely dominates the reward signal.
- **right_lane_reward**: The agent is given a small reward (0.1) for staying in the right lane. This encourages polite driving behavior consistent with many real-world highway rules, where slower vehicles are expected to stay to the right.
- **high_speed_reward**: A higher reward (0.4) is awarded for driving at high speeds. This promotes efficiency and aligns with the goal of timely travel, while also increasing the risk of collisions.
- **lane_change_reward**: No additional reward or penalty is assigned to lane changes. This neutral setting prevents the agent from either favoring or avoiding lane changes unless necessary based on context.

These parameter choices aim to strike a balance between safety, efficiency, and realism, while keeping the learning task challenging and engaging. The reward shaping helps direct the agent toward desirable behaviors without over-constraining exploration.

B. Action Spaces

A key motivation for this study was to compare discrete and continuous control paradigms. DQN and PPO operate in a discrete action space where decisions are limited to a predefined set of actions (e.g., turn left, accelerate). These policies are often easier to train and result in more predictable driving behavior. In contrast, TD3 and SAC are continuous control algorithms. They allow fine-grained control over throttle and steering but are more susceptible to instability and require careful tuning.

C. Policy Architecture (MlpPolicy vs CnnPolicy)

For all algorithms in this project, we used the `MlpPolicy` provided by `stable-baselines3`. The MlpPolicy refers to a fully connected multi-layer perceptron network that maps low-dimensional state observations to actions or distributions over actions. This architecture is well-suited for the default observation space in `highway-env`, which provides structured features such as vehicle positions, velocities, and lane indices.

Mathematically, the MLP-based policy approximates the function:

$$a_t = \pi(s_t) = f_\theta(s_t)$$

where s_t is the state input, a_t is the resulting action, and f_θ represents the learned neural network function parameterized by θ. The policy is composed of sequential layers such that:

$$f_\theta(s_t) = \phi_L(\phi_{L-1}(...\phi_1(s_t W_1 + b_1)...))$$

Each ϕ is a non-linear activation (e.g., ReLU), and W_i, b_i are the weights and biases of each layer. In our setup, we used two hidden layers of sizes 512 and 256.

While the alternative `CnnPolicy` is designed for raw pixel input using convolutional layers, it is more suitable for visual environments where agents learn directly from rendered images. Since `highway-fast-v0` offers high-level numerical observations, an MLP is a more efficient and appropriate choice, resulting in faster training, better interpretability, and lower computational overhead.

In future extensions, using `CnnPolicy` may be explored for agents trained on rendered RGB frames. This could improve generalization and visual robustness, but at the cost of significantly more training time and hardware demand.

V. Reinforcement Learning Algorithms

A. DQN

DQN approximates the Q-function $Q(s, a)$ using a deep neural network and updates it via:

$$Q(s_t, a_t) \leftarrow r_t + \gamma \max_a Q(s_{t+1}, a)$$

It uses experience replay and a target network to stabilize training. Suitable for discrete action spaces.

B. PPO

PPO optimizes a clipped surrogate objective to stabilize learning:

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

Where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$. It is robust and sample-efficient.

C. TD3

TD3 is an actor-critic algorithm for continuous control. It reduces overestimation using twin critics and delays policy updates:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\nabla_a Q(s, a|\theta^Q) \nabla_\theta \mu(s|\theta^\mu)\right]$$

It also applies noise to target actions to smooth updates and reduce variance.

D. SAC

SAC augments the objective with an entropy term to encourage exploration:

$$J(\pi) = \sum_t \mathbb{E}_{(s_t, a_t)} \left[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))\right]$$

It's particularly effective in continuous action spaces where exploration is key to long-term performance.

TABLE I: Complete Hyperparameters Used for Training Each Algorithm

Algorithm	LR	Buffer	Batch	Start	Gamma	Arch	Notes
DQN	5e-4	100k	64	500	0.99	(512, 256)	ε-greedy policy, exploration fraction = 0.2, final ε = 0.05, target update = 100
PPO	2.5e-4	N/A	64	N/A	0.99	Default	n_steps = 2048, n_epochs = 10, clip_range = 0.2, GAE λ = 0.95
TD3	1e-3	100k	64	500	0.99	(400, 300)	τ = 0.005, train_freq = 1, target policy smoothing enabled
SAC	3e-4	100k	64	500	0.99	(400, 300)	τ = 0.005, train_freq = 1, entropy coefficient α auto-tuned

VI. Results and Discussion

A. Metrics Used

- **Average Reward** – Learning performance.
- **Collision Rate** – Safety evaluation.

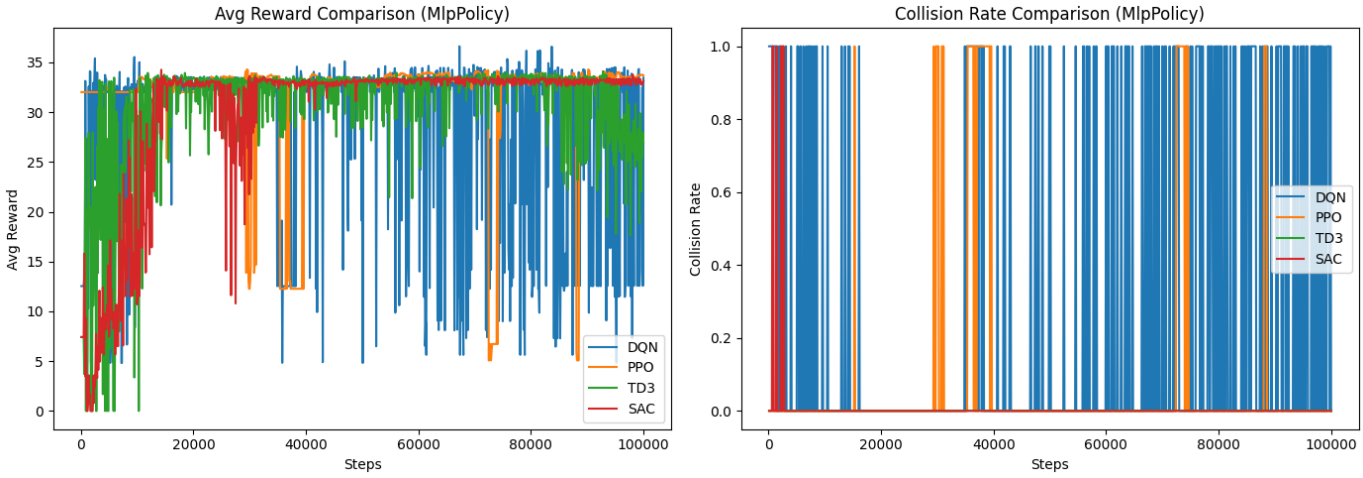

Fig. 2: Rewards and collisions rate for all agents

The metrics were logged onto csv files as the agent trained for each of the algorithms. It was therefore possible to make smoothed out plots of the metrics.

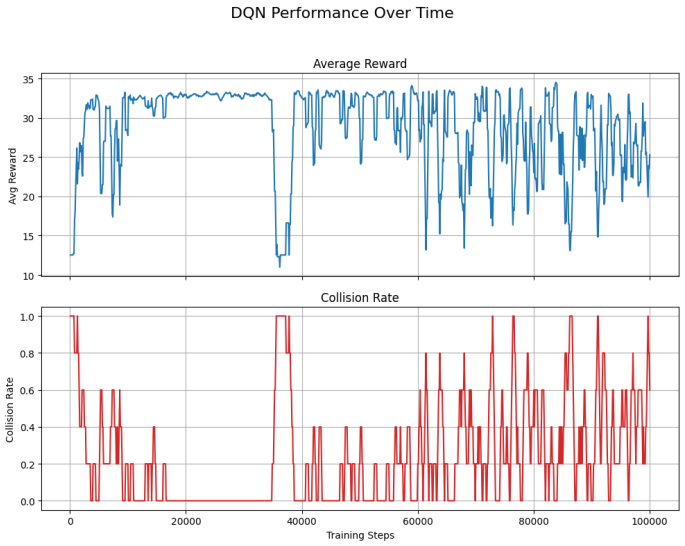

Fig. 3: DQN training performance: Rewards and collision rate

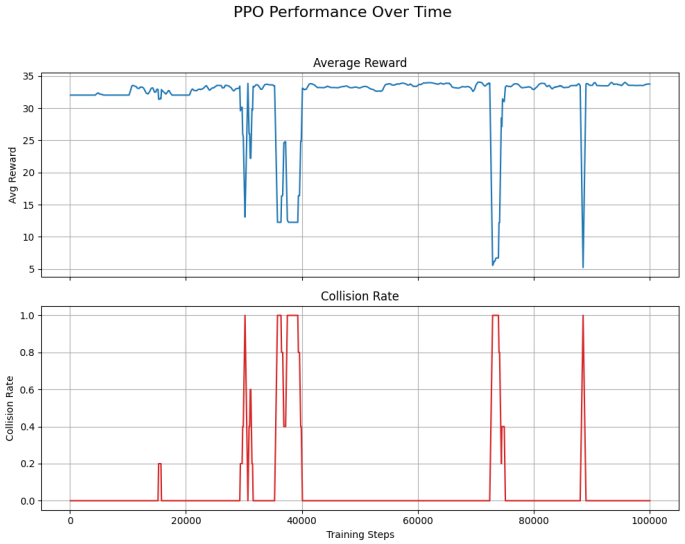

Fig. 4: PPO training performance: Rewards and collision rate

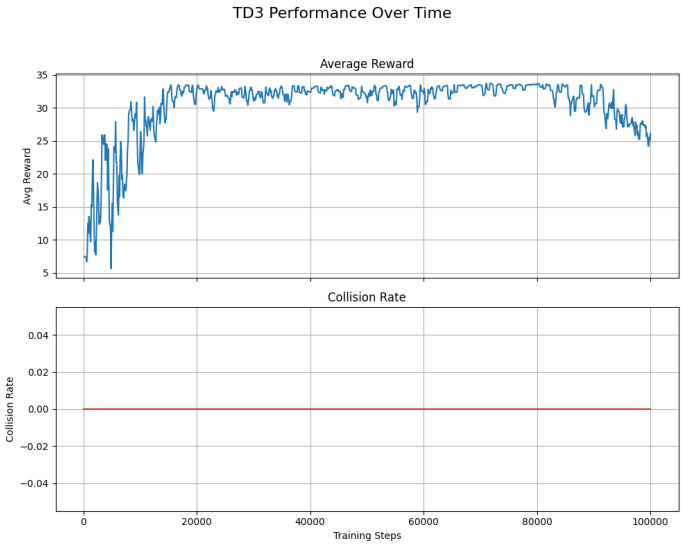

Fig. 5: TD3 training performance: Rewards and collision rate

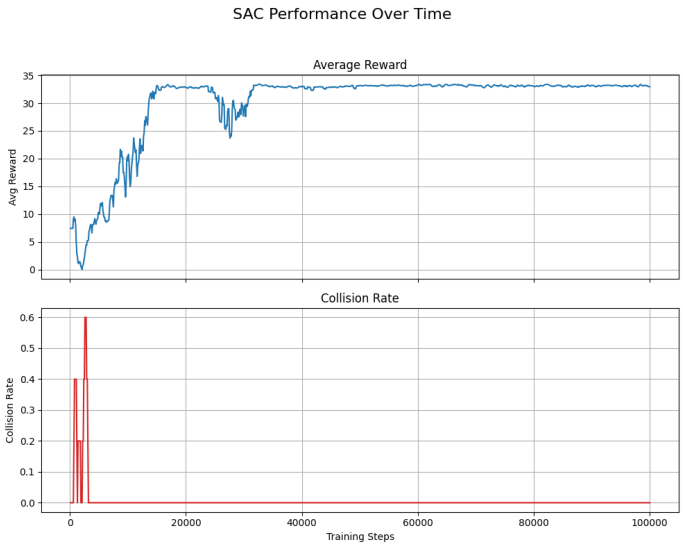

Fig. 6: SAC training performance: Rewards and collision rate

B. Performance

While TD3 had a perfect collision record, it achieved this by drifting off-road, exploiting reward structures that did not penalize leaving the lane. SAC also experienced drifting behavior early in training, but gradually learned to stabilize and remain in-lane. PPO displayed stable and structured lane-keeping throughout training. DQN struggled with consistent lane discipline and had high collision variability.

C. Drifting Behavior

Drifting occurred primarily in TD3 and SAC due to the nature of continuous control. Without discrete boundaries to guide action choices, these agents often issued small but persistent lateral commands, gradually veering out of bounds. This was exacerbated by the reward function not penalizing off-road behavior explicitly. Future implementations should add off-road penalties or use constraint-aware policies (e.g., rule-based post-processing, lane-keeping filters, or Control Barrier Functions) to address this issue and ensure safe, on-road behavior.

VII. Conclusion

This study evaluated four model-free RL algorithms in a highway setting. PPO showed the best trade-off between learning, stability, and realistic behavior. SAC had strong potential, especially after long training. TD3 was highly safe in terms of collision avoidance but produced unrealistic off-road driving behaviors. DQN was the least effective due to unstable learning dynamics. Future improvements should incorporate off-road penalties, constraint-aware training, and possibly hybrid learning strategies to combine RL with imitation or safety filters.

VIII. Improvements and Future Work

While this study demonstrated the strengths and weaknesses of each RL agent in the context of highway driving, several areas offer opportunities for future enhancements:

A. Off-Road Penalty for Continuous Control Algorithms

Both TD3 and SAC occasionally learned to drift off-road to avoid collisions. This behavior, although reducing crash rates, is unrealistic and potentially dangerous. The root cause lies in the reward function's lack of penalties for leaving the road. We recommend modifying the reward function as follows:

$$r_{\text{total}} = r_{\text{base}} + r_{\text{offroad}}, \quad r_{\text{offroad}} = \begin{cases} -2.0 & \text{if agent leaves road bounds} \\ 0 & \text{otherwise} \end{cases}$$

Incorporating this term during retraining can penalize unsafe exploration and encourage proper lane-following behavior in SAC and TD3.

B. Using CnnPolicy for Raw Visual Input

In future experiments, the environment can be reconfigured to provide image-based observations, enabling the use of `CnnPolicy`. Training agents with convolutional networks would allow them to learn directly from visual stimuli, similar to how humans perceive driving scenes.

Comparing `MlpPolicy` and `CnnPolicy` on the same environment will help evaluate trade-offs between efficiency and realism. CNNs may also prove beneficial in transferring skills to real-world driving systems or to other visual simulation environments.

C. Hybrid Learning Approaches

A promising direction would be to combine imitation learning with reinforcement learning. By pretraining agents on expert demonstrations (e.g., human driving data), then refining their policies through interaction, we can potentially speed up training and reduce unsafe early exploration.

D. Safety Constraints and Rule Enforcement

Incorporating safety constraints, such as lane boundaries, speed limits, and rule-based post-processing, can prevent the agent from executing unsafe maneuvers. Techniques like Control Barrier Functions (CBFs) or shielded RL could help ensure compliance with highway norms even during learning.

E. Curriculum and Transfer Learning

Progressively increasing environment difficulty—by starting with fewer vehicles or simpler dynamics—can help agents gradually acquire complex behaviors. This curriculum learning strategy can make training more efficient and reduce convergence time.

F. Evaluation in Diverse Environments

Finally, agents trained in the current setup can be tested in other variations of the highway environment (e.g., intersections, merges, or weather conditions) to assess policy robustness, adaptability, and generalization.

IX. Acknowledgement

Thanks to Ali Jnadi, the assistant instuctor for the reinforcement learning course at Innopolis University for his support, feedback and guidance with this team project.

X. Contributions

Kelvin Asu Ekuri, David Daniel, Danil Afonchikov, and Ivan Kiselyov contributed equally to the design, development, training, and analysis of reinforcement learning agents in this project. The team collaborated on implementing the training pipeline, hyperparameter tuning, and logging. All members contributed to writing, editing, and structuring the report.

We jointly spent approximately 60 hours training agents and experimentation, and over 20 hours writing and reviewing the report. Additional time was dedicated to debugging, exploring alternate algorithm implementations, testing the environment, and researching literature and existing projects.

References

[1] K. Balasubramanian, A. Kothari, and K. Vijayakumar, "Autonomous driving in a multi-lane highway environment," *International Journal of Innovative Technology and Exploring Engineering*, vol. 9, pp. 633–637, 06 2020.

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

[3] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *International Conference on Machine Learning (ICML)*, 2018.

[4] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *International Conference on Machine Learning (ICML)*, 2018.

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

[6] A. Raffin, A. Hill, A. Ernestus *et al.*, "Stable baselines3: Reliable reinforcement learning implementations," 2021, https://github.com/DLR-RM/stable-baselines3.

[7] R. Sinha, J. Sæther, J. Sæther, and T. Veiby, "Racetrack navigation on openai gym with deep reinforcement learning," https://web.stanford.edu/class/aa228/reports/2019/final9.pdf, 2019, stanford AA228 Final Project Report.

[8] A. Sagar, "Deep reinforcement learning tutorial with openai gym," https://towardsdatascience.com/deep-reinforcement-learning-tutorial-with-open-ai-gym-c0de4471f368, 2019, towards Data Science.

[9] F. Foundation, "Highway-env quickstart guide," https://highway-env.farama.org/quickstart/, 2024, accessed April 2025.