The Impact of Computation Delay on Deep Reinforcement Learning in Autonomous Driving

Abstract

Autonomous driving systems rely on real-time decision-making, often facilitated by deep reinforcement learning (DRL). However, real-world deployment introduces computation delays due to sensor processing, neural network inference, and network communication latency. These delays can significantly impact vehicle performance, leading to unsafe behaviours such as delayed braking or late obstacle avoidance. In this study, we explore how computation delay affects DRL-based agents in highway driving simulations using the *highway-env* framework. We evaluate performance in *highway-v0*, which uses discrete actions, and our modified *highway-v1*, which improves continuous action control by refining the reward function and ensuring stable forward motion. To understand the impact of delays, we test different levels of computation delay on DRL algorithms like DQN, PPO, and DDPG. Our results show that increasing delay negatively impacts performance, with discrete-action policies being more sensitive than continuous-action ones. To support further research in delay-aware reinforcement learning, we also introduce a custom wrapper that adds configurable computation delays to any OpenAI Gym environment.

Introduction

Recent advancements in autonomous driving technology have enabled real-world applications ranging from driver-assist systems to fully self-driving vehicles. These systems rely on deep reinforcement learning (DRL) to make decisions in dynamic traffic environments. However, real-world deployment is hindered by **computation delays**, arising from sensor processing, model inference, and network latency. Even minor delays can lead to hazardous situations, such as failing to react in time to a pedestrian or delaying a braking manoeuvre, increasing the risk of collisions.

To better understand the effects of computation delay on DRL-based autonomous driving, we conduct a systematic evaluation in a simulated highway environment. Specifically, we analyse agent performance in highway-v0, which uses discrete action spaces, and our custom highway-v1, which extends the environment to continuous action spaces while retaining discrete action compatibility. By introducing different levels of computation delay, we analyze how different DRL algorithms—DQN, PPO, and DDPG—handle delayed execution and compare their robustness.

Our contributions in this study include:

1. **Introducing computation delay** in highway-v0 and highway-v1 and evaluating its effects on DRL-based autonomous driving.
2. **Testing and comparing DQN, PPO, and DDPG** under both delayed and non-delayed conditions.
3. **Visualizing the impact of delays** through performance graphs, including total rewards over episodes and key metrics like mean episode length and crash rate.
4. **Analysing exponential computation delay** to simulate real-world unpredictable latencies.
5. **Providing an open-source implementation**, including a custom wrapper that adds configurable computation delays to any OpenAI Gym environment, supporting further research in delay-aware reinforcement learning.

By systematically studying computation delay, this research offers insights into the robustness of DRL-based autonomous driving models and emphasizes the need for delay-aware learning strategies in real-world deployment.

Background and Related Work

Highway-env simulator

The highway-v0 environment is a commonly used reinforcement learning environment designed to simulate highway driving scenarios. It provides a discrete action space, where the agent selects from predefined actions such as changing lanes, accelerating, and braking. This environment facilitates research on decision-making in multi-agent traffic scenarios, requiring the ego vehicle to navigate efficiently while optimizing performance metrics including velocity, fuel efficiency, and safety.

A significant limitation of highway-v0 is its reliance on discrete actions, which simplifies control but may not reflect real-world autonomous driving requirements where steering and acceleration are continuous. To address this,

we modified the highway-v0 to highway-v1 environment to improve continuous action control by refining the reward function. The primary modification in highway-v0 is an updated reward function that encourages the vehicle to move forward. This change was necessary because in its default form, when switching to a continuous action space, the vehicle initially exhibited unstable behaviour, such as moving off the road or looping in place. However, in the discrete action setting, the agent already moved forward correctly.

Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) integrates deep learning with reinforcement learning (RL), enabling agents to learn from complex, high-dimensional data. In traditional RL, an agent interacts with its environment by selecting actions that maximize cumulative rewards. However, when dealing with large state spaces, conventional RL methods face challenges, as storing and updating Q-values for every possible state becomes impractical.

To address this, DRL leverages deep neural networks (DNNs) to estimate value functions or policies. These multi-layered networks process raw input data—such as images or sensor readings—to extract meaningful features and guide decision-making. This capability allows DRL agents to operate effectively in complex environments, including autonomous driving and robotic systems.

Despite its advantages, DRL is highly sensitive to real-time constraints like computation delays, which can negatively impact performance in autonomous systems. If a vehicle processes delayed sensor data, it may struggle to react appropriately to critical situations, increasing the risk of unsafe behaviour. Therefore, analysing DRL performance under delayed conditions is crucial for real-world applications like self-driving cars.

Deep Reinforcement Learning Algorithms

Deep Q-Network (DQN)

Deep Q-Network (DQN) is a value-based reinforcement learning algorithm introduced by (Mnih et al. 2015) [2] that extends Q-learning with deep neural networks. Instead of using a Q-table, DQN approximates the Q-value function Q(s,a;θ) using a neural network, where θ represents the model parameters. The Q-values are updated based on the Bellman equation:

$$Q(s,a) = Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

where α is the learning rate, γ is the discount factor, r is the immediate reward, s' is the next state, and a' is the next action chosen optimally.

To improve stability, DQN incorporates two key techniques:

- **Experience Replay** – Stores past experiences in a memory buffer and randomly selects them during training. This helps break the correlation between consecutive updates, making learning more stable.
- **Target Network** – Uses a separate Q-network to calculate target values. This network is updated at regular intervals, preventing drastic changes in learning and improving stability.

DQN follows an epsilon-greedy exploration strategy, where the agent takes random actions with probability ϵ and exploits learned policies otherwise. Since DQN operates in a discrete action space, it is well-suited for highway-v0.

Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is a policy-gradient-based reinforcement learning algorithm that directly optimizes the policy πθ(a|s) instead of estimating Q-values [3]. PPO is an **actor-critic** method, where:

- The **actor** learns the policy πθ(a|s), which maps states to actions.
- The **critic** evaluates the chosen actions using a value function V(s)V(s)V(s), which estimates future rewards.

To ensure stable policy updates, PPO introduces a clipped objective function that prevents drastic policy changes:

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

where rt(θ) is the probability ratio of new and old policies, At is the advantage function, and ϵ is a small constraint for stable updates.

PPO is suitable for both discrete and continuous action spaces, making it effective in highway-v0 and highway-v1. Unlike DQN, PPO is more resilient to delays, as it learns a probability distribution over actions rather than relying on immediate Q-values.

Deep Deterministic Policy Gradient (DDPG)

Deep Deterministic Policy Gradient (DDPG) is an actor-critic reinforcement learning algorithm for continuous action spaces, making it ideal for highway-v1. It extends

Deterministic Policy Gradient (DPG) using deep neural networks [4].

- The **actor network** $\pi\theta(s)$ directly outputs continuous actions.
- The **critic network** $Q\phi(s,a)$ estimates Q-values to guide the actor.

DDPG improves stability using:

- **Experience Replay** – Stores past experiences to break correlation in training.
- **Target Networks** – Uses separate target networks for the actor and critic to stabilize learning.

The critic is trained using the Bellman equation:

$$L(\phi) = \mathbb{E}\left[(r + \gamma Q_{\phi'}(s', \pi_{\theta'}(s')) - Q_{\phi}(s, a))^2\right]$$

DDPG is highly effective for continuous control, but it is highly sensitive to delays. Since continuous actions require fine adjustments, delayed actions often result in loss of control, leading to higher crash rates in our experiments.

Computation Delay in Real-World Applications

In real-world autonomous driving, computational constraints introduce various forms of latency. Sensor processing, network communication, and inference computation generate delays between perception and action execution. Understanding how reinforcement learning agents perform under such delays is crucial for developing robust autonomous systems.

Reinforcement Learning with Random Delays

Recent research, such as the study *"Reinforcement Learning with Random Delays"* (ICLR 2021) [5], has explored how RL agents perform under stochastic delays. The paper introduces the concept of Random Delay Markov Decision Processes (RDMDP), which augment the state space with past actions and delay information. It demonstrates how naive RL approaches can struggle with credit assignment in delayed environments, leading to degraded performance.

In real-world applications, real-time deep reinforcement learning is essential for autonomous systems that must make decisions under strict time constraints. Computation delays in sensor processing, model inference, and control execution introduce latency, which can lead to poor decision-making and unsafe behaviours. Traditional DRL models assume near-instantaneous action execution, but real-time constraints demand adaptive strategies to handle varying delay conditions.

To address these issues, the study presents a **Delay-Correcting Actor-Critic (DCAC)** method that improves policy learning by correcting for delay-induced bias. This aligns with our findings that increasing computation delays negatively impact performance in highway-v0 and highway-v1. Future work could explore integrating real-time DRL techniques, such as delay-aware policy optimization or model-based prediction, to mitigate these effects in autonomous driving environments.

Methodology

Custom highway-v1 implementation

The highway-v1 environment improves upon highway-v0 by refining continuous action execution. While highway-v0 inherently supports both discrete and continuous actions, highway-v1 enables smoother acceleration and steering while ensuring stable forward motion through an improved reward function. This was achieved by modifying the action representation, vehicle dynamics, and reward structure while retaining the multi-agent traffic interaction framework. Additionally, highway-v1 retains discrete action support, ensuring compatibility with both types of control strategies.

Key Features of highway-v1

- Highway-v1 allows smooth control over acceleration and steering.
- Adjustments in vehicle initialization, lane density, or termination conditions.
- New constraints include penalties for stopping or reversing to encourage forward movement and safe driving behaviour
- **Computation Delay Handling**: To simulate real-world processing delays, highway-v1 implements an *elapse()* function that delays the execution of the agent's action while the environment and other vehicles continue updating. The delay duration is determined by the simulation frequency, meaning that for longer delays, more timesteps pass before the agent executes its next action. This method ensures that the model experiences realistic latency effects, such as those caused by sensor processing or network communication in real-world autonomous systems

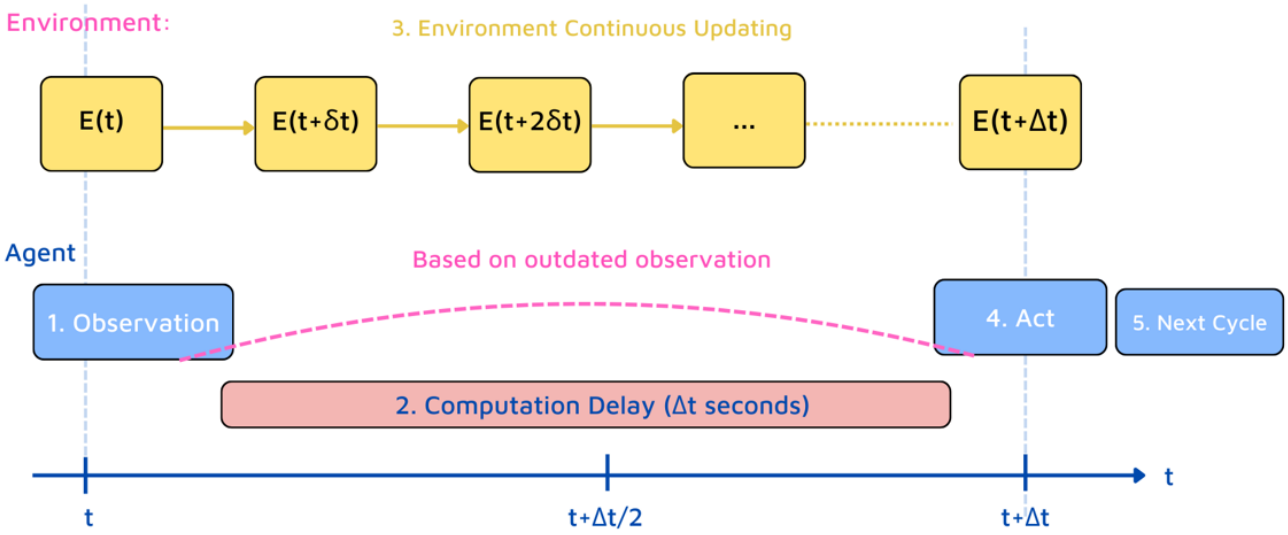


Figure: Agent-Environment Interaction with Computation Delay

The figure above illustrates the effect of computation delay in reinforcement learning-based autonomous driving. At each timestep t, the agent receives an observation from the environment. However, due to computational processing time (Δt), the agent's action is executed based on outdated information. Meanwhile, the environment continues updating in real-time, potentially making the agent's decision suboptimal. This delay can lead to unstable control, increased crash rates, and degraded performance, particularly in continuous action spaces. To simulate this effect, we introduce an elapse() function in highway-v1, which imposes a controlled delay before executing the agent's chosen action

These modifications enable highway-v1 to simulate a more realistic autonomous driving scenario where actions are continuous, and decision delays can be explicitly modelled. The highway-v1 environment was modified to support continuous action spaces, allowing for finer control over acceleration and steering. This was achieved by adapting the action representation and modifying the environment dynamics accordingly.

Computation Delay Setup

We introduced computation delays in highway-v0 using an ElapseActionWrapper, which ensures that an agent's action is executed only after a predefined delay period. The delay was randomly sampled from an exponential distribution within the range of [0,2] seconds.

To better analyse the effects of significant computation delays, we set the mean of the exponential delay distribution to **1.2 seconds**. This allows us to observe how the model handles larger, more realistic delays that could occur due to sensor processing, network latency, or computational overhead in real-world autonomous driving systems

Exerimental Setup

- **Algorithms:** DQN, PPO and DDPG.
- **Environment Configurations:**
  - highway-v0: Discrete actions.
  - highway-v1: Discrete and Continuous actions.
- **Evaluation Metrics:**
  - Total episode reward.
  - Episode length (number of timesteps before termination).
  - Crash rate.
  - Performance after delays comparisons across different DRL models.

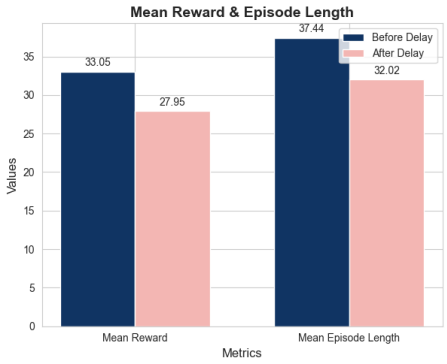
Results and Analysis

To assess the impact of computation delays, we compared the performance of different DRL algorithms in both highway-v0 and highway-v1. The evaluation included:

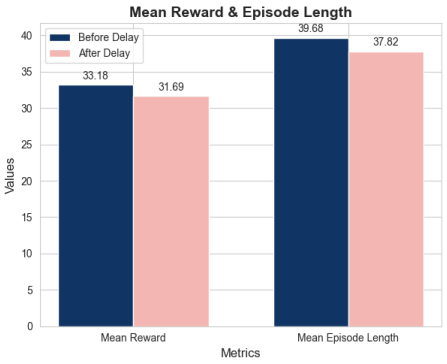
- **Baseline Performance:** Testing each model **without computation delay** to establish a reference point.
- **Exponential Computation Delay:** Introducing random delays sampled from an exponential distribution, simulating real-world variations in processing time

Bar Chart Comparisons

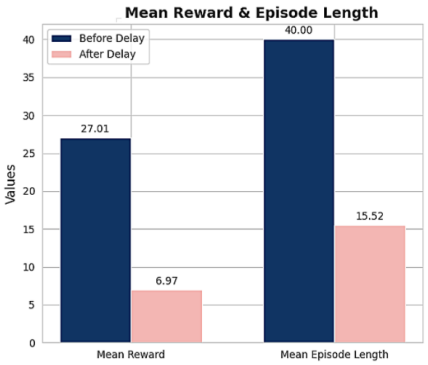
Highway v1:



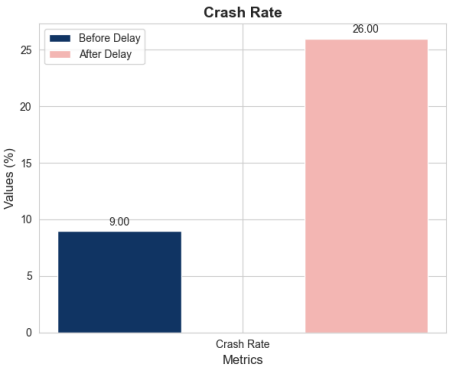
discrete DQN



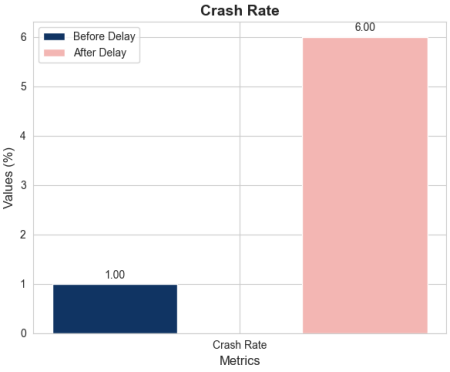
discrete PPO



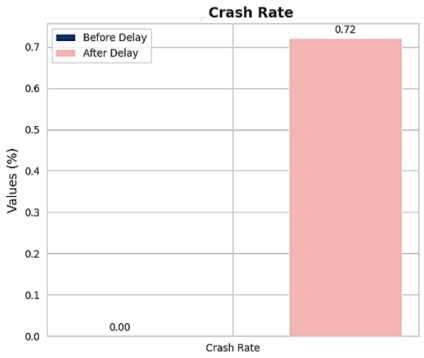
continuous DDPG



discrete DQN

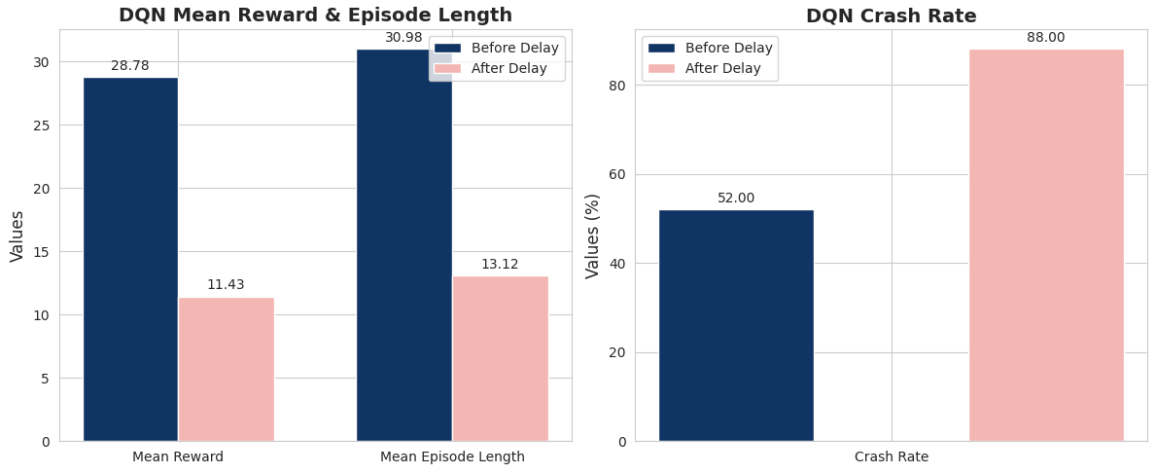


discrete PPO

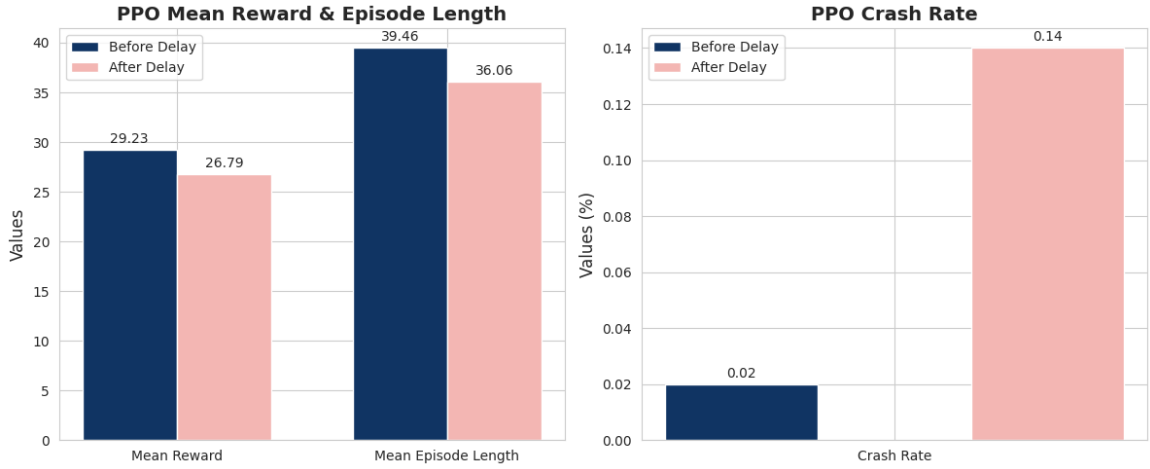


continuous DDPG

Highway v0:



discrete DQN



discrete PPO

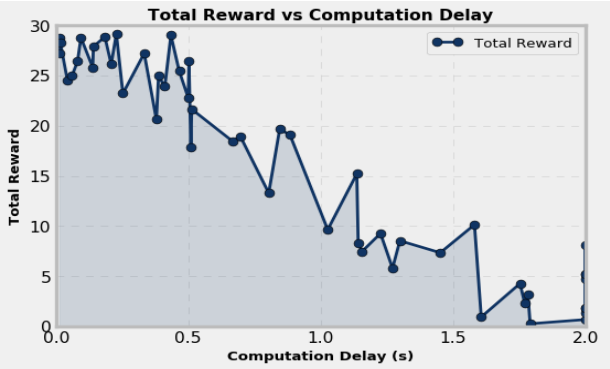
The figure illustrates the performance of each model over 50 test episodes. In the no-delay scenario, all models illustrate strong performance with minimal crashes. However, when computation delay is introduced, the agents struggle to react in time, leading to a significant increase in crash rates.

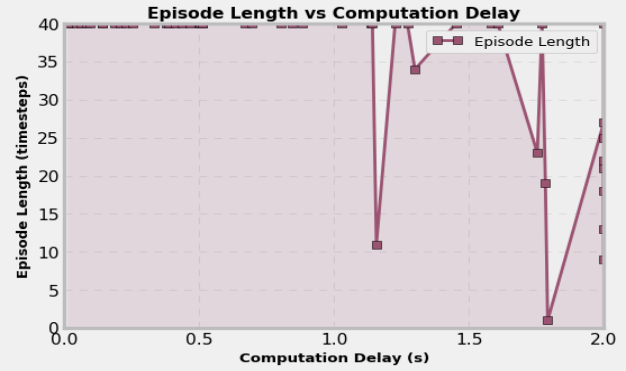
Notably, DDPG outperforms discrete-action models (DQN and PPO) in the no-delay scenario, achieving zero crashes while maintaining smooth driving. However, when delay is applied, DDPG's performance deteriorates dramatically, leading to a much higher crash rate compared to DQN and PPO. This suggests that continuous control strategies may be more sensitive to delays, since they depend on precise, frequent adjustments that lose effectiveness when there's a timing lag.

The second figure, a bar chart comparing mean episode reward, episode length, and crash rate, further emphasizes the negative impact of delay across all models, with DDPG experiencing the most severe degradation in performance.

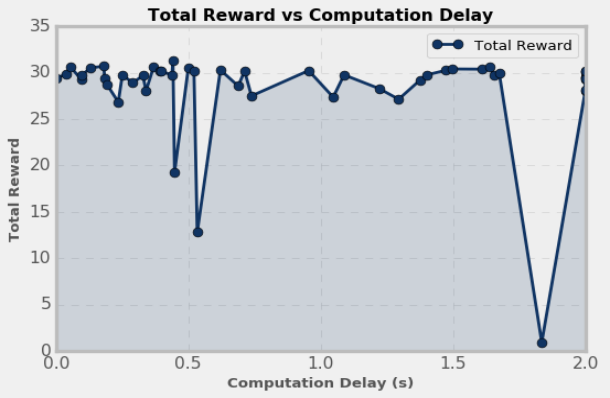
Exponential Computation Delay Impact

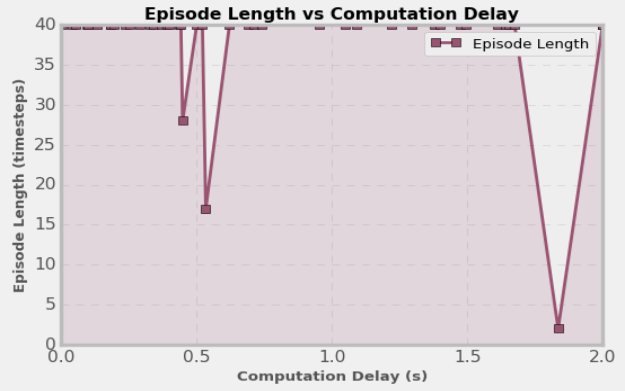
In this section, we analyse the effect of exponentially distributed computation delays on the performance of DDPG and PPO. Since continuous action spaces more accurately reflect real-world autonomous driving scenarios than discrete action spaces, DDPG serves as a valuable benchmark for understanding how computation delays affect real-world driving models.





Continuous DDPG





Discrete PPO

With an average delay of 1.2 seconds, the figure illustrates how DDPG functions at various delay levels, ranging from 0 to 2 seconds. The findings suggest that while DDPG can adjust to little delays, it suffers greatly from longer delays. The model maintains a low crash rate and obtains significant rewards when delays are small. Its capacity to make prompt decisions, however, deteriorates with increasing time, leading to more crashes and lower rewards. This demonstrates how crucial computation delay is to autonomous driving, especially for continuous-action models like DDPG, where timely and accurate control is crucial for efficiency and safety.

Unlike DDPG, PPO's performance did not show a clear downward trend. Instead, the total reward fluctuated randomly at different delay levels. One reason is that PPO uses discrete actions (e.g., lane changes, acceleration choices) instead of continuous control, making it less sensitive to short delays. Another reason is that the environment might be simple enough that PPO can still perform reasonably well despite delays. However, PPO still had more crashes compared to the no-delay case, meaning delay still had an impact, just not in a way that got worse over time.

Overall, continuous models like DDPG suffer more from increasing delay, while discrete models like PPO are more stable but still experience more crashes when delay is present. This highlights the need to design reinforcement learning models that can handle real-world delays effectively.

Discussion

The findings in this study highlight the critical impact of computation delay on reinforcement learning agents in autonomous driving. Across all models tested, introducing delays led to a notable decline in performance, with increased crashes and instability.

In a no-delay setting, DDPG outperformed discrete models (DQN, PPO) by enabling smoother and more adaptive control. However, when delays were introduced, DDPG suffered the most, experiencing a sharp increase in crashes and unstable behaviour. This is because DDPG relies on continuous, fine-grained adjustments, which become ineffective when execution is delayed.

The exponential delay study reinforced the observation that computation delay disproportionately affects continuous control models. This finding is especially important for real-world applications, where autonomous vehicles need to make low-latency decisions to ensure safety. If an agent can't react quickly to changing traffic conditions, it could lead to dangerous situations like sudden braking failures or collisions.
Overall, our results highlight that real-time decision-making is a major challenge in reinforcement learning for autonomous driving and that computation delay needs to be explicitly considered in future models.

Conclusion

In this paper titled "The Impact of Computation Delay on Deep Reinforcement Learning in Autonomous Driving," we examined how computation delays influence both discrete and continuous action models within reinforcement learning (RL) frameworks for autonomous driving. To facilitate this investigation, we created highway-v1, an enhanced version of highway-v0 that allows for continuous control to better simulate realistic vehicle dynamics. Our findings indicate that computation delays have a detrimental effect on all models, resulting in increased crash rates and diminished overall performance. Although DDPG excelled in scenarios without delays, it showed a high degree of sensitivity to delays, making it less suitable for practical applications. Conversely, the performance of PPO was more erratic when faced with delays, highlighting that discrete models might react differently to delays compared to continuous ones, particularly in less complex environments.

By simulating exponentially distributed delays, we demonstrated that variable delays exacerbate performance issues, underscoring the necessity for reinforcement learning strategies that take delays into account. These results underline the importance of addressing computation delay to enhance the robustness and safety of RL-based autonomous systems. Given that real-world scenarios often involve sensor, network, and processing delays, the creation of delay-compensated models will be essential for the safe implementation of autonomous vehicles.

As part of our research, we are also offering an open-source implementation that includes a delay wrapper for OpenAI Gym environments. This resource aims to facilitate further experimentation in delay-aware reinforcement learning, contributing to the advancement of more resilient and adaptive RL models for autonomous driving.

Future Work

To improve delay robustness in reinforcement learning-based autonomous driving, future research should explore:

1. Testing Alternative RL Algorithms in Highway-v1: Investigating other deep reinforcement learning models (e.g., SAC, TD3) in highway-v1's continuous action space to evaluate their robustness under computation delays.
2. Expanding to Complex Environments: Testing RL models in more challenging environments, such as intersections and merging scenarios.
3. Improving Model Performance: Refining training strategies to increase model robustness and ensure more accurate experimental testing.
4. Delay-Compensated Reinforcement Learning: Implementing Delay-Correcting Actor-Critic (DCAC) or other delay-aware RL techniques to enhance resilience against computation lag.
5. Latency-Aware Reward Functions: Developing dynamic reward mechanisms that penalize delayed reactions and encourage proactive decision-making.

By integrating delay-aware techniques, future studies can enhance the reliability and safety of reinforcement learning-based autonomous driving, bringing RL models closer to real-world deployment.

Reference

[1] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, “An Introduction to Deep Reinforcement Learning,” *Foundations and Trends® in Machine Learning*, vol. 11, no. 3–4, pp. 219–354, 2018, doi: https://doi.org/10.1561/2200000071.

[2] V. Mnih *et al.*, “Human-level Control through Deep Reinforcement Learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: https://doi.org/10.1038/nature14236.

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. OpenAI, “Proximal Policy Optimization Algorithms,” Aug. 2017. Available: https://arxiv.org/pdf/1707.06347

[4] T. P. Lillicrap *et al.*, “Continuous control with deep reinforcement learning,” 2015. https://arxiv.org/abs/1509.02971

[5] Y. Bouteiller, S. Ramstedt, G. Beltrame, C. Pal, and J. Binas, “Reinforcement Learning with Random Delays,” *OpenReview*, 2021. https://openreview.net/forum?id=QFYnKlBJYR (accessed Mar. 05, 2025).