Traffic optimization in a simple network using Deep Reinforcement Learning

Volodymyr Levytskyi[1,†], Oleksii Lopuha [1,†] and Pavlo Kruk [1,†]

[1] *Kyiv National University of Construction and Architecture, 31, Air Force Avenue, Kyiv, 03037, Ukraine*

Abstract

The optimization of traffic flow in urban environments remains a major challenge in contemporary research, despite the substantial body of scientific literature dedicated to this issue. While significant progress has been made, a universally effective solution applicable to real-world conditions has yet to be developed. One of the primary challenges lies in processing the vast amounts of incoming traffic data continuously collected from sensors distributed across urban road networks. Traditionally, due to the scale of this task, researchers have concentrated on designing systems with localized agents. These agents typically manage traffic at individual intersections, with coordination occurring within multi-agent systems. Recent advancements, however, address the complexity and volume of input data through the integration of deep learning techniques. In particular, the deep deterministic policy gradient (DDPG) algorithm has been proposed as a means to process extensive traffic data efficiently. An experimental study was conducted using a simplified intersection model to assess the effectiveness of this approach. The results indicate that the DDPG algorithm outperformed Q-learning in this scenario, achieving a reward in the range of 4 to 4.3 points, compared to Q-learning's reward range of 2 to 4 points. Performance evaluation, based on the average reward per episode, revealed that while both DDPG and Q-learning attained comparable reward levels, DDPG demonstrated more stable convergence (0.04–0.21 points), whereas Q-learning exhibited greater instability (0.04–0.43 points). Furthermore, an analysis of intra-episode performance indicated that DDPG achieved performance gains primarily toward the latter stages of each episode. Overall, the algorithm proved to be effective within this experimental framework, and the findings provide a foundation for further enhancements and applications in more complex traffic management scenarios.

Keywords

DDPG, Aimsun, Q-learning, road traffic

1. Introduction

Urban planning has evolved from a pedestrian-centric approach to the development of complex transportation hubs that accommodate not only sidewalks and roadways but also tram lines and metro stations. This transformation has introduced substantial challenges in traffic management, necessitating the implementation of various regulatory tools, including traffic signals, traffic lights, and systematic transit planning, to ensure efficient urban mobility.

Modern traffic lights operate under two principal modes: fixed-time programs and actuated systems. Fixed-time programs also referred to as pre-planned controls, allocate predetermined

[1]*ITTAP'2025: 5th International Workshop on Information Technologies: Theoretical and Applied Problems, October 22-24, 2025, Ternopil, Ukraine, Opole, Poland*

* Corresponding author.

† These authors contributed equally.

levitscyi@gmail.com (V. Levytskyi); alekseylopuga@gmail.com (O. Lopuha); kruk97@ukr.net (P. Kruk)

0000-0003-1829-488X (V. Levytskyi); 0000-0001-6397-2710 (O. Lopuha); 0000-0002-6786-452X (P. Kruk)

durations for red, yellow, and green phases, regardless of real-time traffic conditions. In contrast, actuated traffic lights dynamically adjust their signal phases based on data collected from local traffic sensors installed near intersections. While this adaptive mechanism improves traffic flow at individual intersections, it can hinder synchronization with adjacent intersections, reducing its effectiveness in densely populated urban settings [1, 2].

Furthermore, existing systems do not fully integrate city-wide traffic flow data. Although networked vehicle technologies have the potential to anticipate congestion, their practical implementation is often restricted to routine interventions, such as manual traffic direction by law enforcement. As a result, advanced strategies that could leverage comprehensive traffic data to optimize traffic light performance remain largely underutilized.

2. Literature review

Machine learning presents a significant opportunity to improve traffic control systems by optimizing signal timing based on real-time traffic conditions. Numerous studies have explored the potential of machine learning in this domain. One notable example is a reinforcement learning system developed using the Green Light District simulator, which serves as a foundation for further research in this field [3]. Additional methodologies include fuzzy logic and multi-agent systems, where agents managing individual intersections either exchange information or respond to collective data from connected vehicles [4, 5]. However, these approaches are often limited to isolated intersections or small networks, as they rely on partial traffic data and fail to fully exploit the breadth of available information.

A promising strategy to overcome these limitations is deep reinforcement learning (DRL), a class of algorithms that leverage deep neural networks as function approximators for value functions. The rise of DRL is largely attributed to the success of Deep Q-Networks (DQN), which demonstrated exceptional performance in playing Atari games by processing raw pixel inputs [6].

Within the DQN framework, a neural network receives the state of the environment as input and generates Q-values for all possible actions as output. This process is guided by a loss function that determines the gradient direction. The initial success of reinforcement learning using neural networks for function approximation can be traced back to TD-Gammon [7]. However, despite early enthusiasm, this approach proved ineffective when applied to a wider range of problems, ultimately leading to its decline [8]. The key challenges contributing to its limitations include:

- The neural network was trained on values generated in real-time, resulting in sequential data that exhibited strong correlations with previous values, thereby violating the assumption of independent and identically distributed (i.i.d.) data [9].
- Policy fluctuations arose due to minor variations in Q-values, which in turn affected the distribution of training data.
- Large optimization steps occurred when substantial rewards were received, leading to instability in learning.

To address these stability issues, the authors in [10] proposed several improvements, including:

- *Experience replay*: This technique involves storing past actions and rewards in memory, allowing the neural network to be trained on randomly sampled experiences rather than sequential real-time data. This mitigates the issue of temporal autocorrelation and takes advantage of the off-policy nature of Q-learning.
- *Reward scaling*: By constraining reward values within the range of [−1, +1], this approach prevents excessive weight magnification during error backpropagation.
- *Target network*: Two DQNs are utilized, where one computes target values while the other accumulates weight updates, which are periodically transferred to the first network. This mechanism helps stabilize policy updates by reducing oscillations in Q-values.

Despite these advancements, DQNs are primarily suited for discrete action spaces, making them less effective for continuous action environments such as traffic light control, where scalability remains a major challenge. To address this limitation, the Deep Deterministic Policy Gradient (DDPG) algorithm has been introduced [11]. As its name suggests, DDPG integrates the classical actor-critic framework of reinforcement learning with a deterministic policy gradient [12].

The foundational policy gradient algorithm was originally formulated in [10], where the policy gradient theorem for stochastic policies was established. This theoretical groundwork has since facilitated the development of more advanced methods for optimizing traffic control systems in complex urban environments.

3. Purpose of the work

This study aims to empirically validate the effectiveness of the DDPG algorithm in optimizing traffic flow management in urban environments. The primary focus is on evaluating the performance of the proposed DDPG approach relative to Q-learning using a simulated intersection model. Furthermore, the study examines the progression of both the DDPG and Q-learning algorithms within this simulation over time, providing a comparative analysis of their evolution and effectiveness.

4. Traffic simulation concepts

To assess the effectiveness of this approach, a traffic simulation tool is employed. Specifically, Aimsun [13], a third-party microscopic traffic simulation software, has been selected to model various traffic scenarios, including road networks, intersections, and traffic signals.

A fundamental component of any traffic simulation is the network, which represents roadways and intersections where vehicles navigate. Certain roads are linked to centroids that function as sources and sinks for vehicles. The number of vehicles generated or absorbed by these centroids is determined by an origin-destination demand matrix, in which each cell represents the flow of vehicles between a specific origin and destination centroid. Different demand matrices can be applied at various time intervals during the simulation to simulate real-world traffic dynamics accurately.

Road networks in the simulation often incorporate motion detectors that emulate induction loops embedded in the roadway surface. These detectors collect essential traffic data, including vehicle counts, average speeds, and occupancy rates.

Traffic signals regulate vehicle movement at intersections, ensuring an orderly flow of traffic and preventing congestion. Signals at a given intersection are coordinated so that one direction receives a green light while the opposing direction sees a red light. This synchronization facilitates efficient traffic management. The collective state of all signals at an intersection over a defined period is referred to as a phase, which is determined by both the signal states and their durations. The sequence of these phases forms a control plan, which is executed cyclically over time. To further optimize traffic flow and minimize delays, control plans at adjacent intersections are often synchronized.

Once the simulation begins, centroids release vehicles according to the demand matrix, directing them toward their respective destination centroids. To ensure realistic conditions, a warm-up period is typically implemented, during which centroids generate vehicles to populate the network before the official commencement of the simulation.

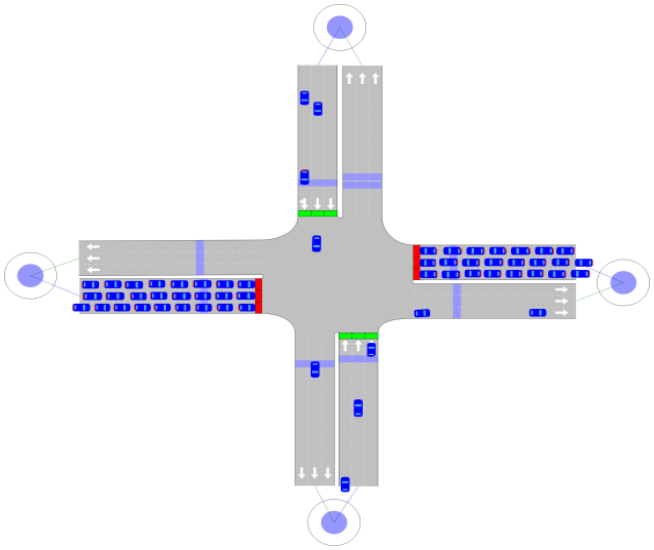

Figure 1: Simple traffic network in Aimsun.

Figure 1 illustrates a simplified transportation network modeled in Aimsun, featuring a single intersection. In this model, vehicles stop at red signals and proceed on green signals, and centroids continuously generate or absorb traffic. Traffic detectors positioned at roadway entry and exit points collect data throughout the simulation, providing valuable insights into traffic dynamics and system performance.

5. DDPG та Q-learning

Most reinforcement learning (RL) algorithms are grounded in the Bellman equation:

$$V^{\pi}(s)=R(s,\pi(s))+\gamma\sum_{s'}P(s'\vee s,\pi(s))V^{\pi}(s'), \quad (1),$$

which defines the expected long-term reward — denoted as the value function V — for taking an action determined by a given policy π, based on the immediate reward *R(s, π(s))* received in state *s*. The discount factor γ (where $0 \leq \gamma < 1$) regulates the relative importance of immediate versus future rewards [14]. The term $\sum_{s'} P(s' \vee s, \pi(s)) V^{\pi}(s')$ represents the weighted sum of the values of possible future states s', where each weight corresponds to the probability of transitioning from state s to s' after executing action $\pi(s)$. Here, $V^{\pi}(s')$ reflects the expected cumulative reward achievable from future state s' under policy π.

Deep Deterministic Policy Gradient (DDPG) is a non-parametric (NP) algorithm that integrates key elements of Q-learning — specifically, utility function approximation — with the actor-critic architecture of reinforcement learning.

Theorem 1 (Policy Gradient). For any Markov Decision Process (MDP), if the policy parameters θ are updated in proportion to the gradient of the performance measure ρ, then θ is guaranteed to converge to a locally optimal policy to ρ. The policy gradient is computed as follows:

$$\Delta\theta \approx \alpha \frac{\partial \rho}{\partial \theta} = \alpha \sum_{s} d^{\pi}(s) \sum_{a} \frac{\partial \pi(s,a)}{\partial \theta} Q^{\pi}(s,a), \tag{2}$$

where α is a positive step size, and $\frac{\partial \rho}{\partial \theta}$ — represents the gradient of the policy performance to the parameters θ. The term $\sum_{s} d^{\pi}(s)$ — denotes the weighted influence of states, where d^{π} is defined as the discounted visitation distribution over states encountered when starting from an initial state s_0 and following policy π: $d^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t P(s^t = s \vee s_0, \pi)$, where γ is the discount factor that determines the present value of future rewards. The term $\sum_{a} \frac{\partial \pi(s,a)}{\partial \theta}$ refers to the gradient of the policy $\pi(s,a)$, indicating how changes in the parameters θ influence the probability of selecting action a in state s. Finally, $Q^{\pi}(s,a)$ is the action-value function, which represents the expected cumulative reward when taking action a in state s and subsequently following policy π [10].

This theorem was further extended in the same paper to accommodate cases where an approximation function f is used in place of the policy π, as formalized in **Theorem 2**:

Theorem 2 (Policy Gradient with Function Approximation). The policy gradient theorem holds when the policy π is represented by an approximation function $f(s,a;w)$ provided that the updates to the weights w diminish as the function f converges to the true policy π. Specifically, the gradient of the expected return to the parameters w can be expressed as:

$$\sum_{s} d^{\pi}(s) \sum_{a} \pi(s,a) [Q^{\pi}(s,a) - ¿ f(s,a;w)] \frac{\partial f(s,a;w)}{\partial w} = 0 ¿, \tag{3}$$

$\left[Q^{\pi}(s,a)-f(s,a;w)\right]$ represents the temporal difference error, which quantifies the discrepancy between the true action-value function $Q^{\pi}(s,a)$ and its approximation $f(s,a;w)$. This error guides the optimization process by indicating how far the current estimate f deviates from the actual value under policy π.

The term $\frac{\partial f(s,a;w)}{\partial w}$ denotes the gradient of the approximating function to its parameters w. This gradient reflects how changes in the parameters affect the estimated value for a given state-action pair and is used during the update step to minimize the approximation error. Together, these elements are central to the learning process in actor-critic and function-approximation-based reinforcement learning algorithms, where the goal is to iteratively adjust w to reduce the gap between the estimated and actual value functions.

If the function f is compatible with the policy parameterization — meaning it satisfies the condition: $\frac{\partial f(s,a;w)}{\partial w}=\frac{\partial \pi(s,a)}{\partial \theta}\frac{1}{\pi(s,a)}$, where $\frac{1}{\pi(s,a)}$ serves as a scaling factor accounting for the inverse probability of selecting action a in state s, then the policy gradient can be simplified as: $\frac{\partial \rho}{\partial \theta}=\sum_{s} d^{\pi}(s)\sum_{a}\frac{\partial \pi(s,a)}{\partial \theta} f(s,a;w)$ [10]. This result is foundational in compatible function approximation, as it allows the use of function approximators (e.g., neural networks) in policy gradient methods while preserving convergence guarantees under certain conditions.

To enhance the stability of the Deep Deterministic Policy Gradient (DDPG) algorithm, the same stabilization techniques used in Deep Q-Networks (DQNs) can be employed — namely, reward normalization, experience replay, and the incorporation of a separate target network. For the latter, DDPG integrates two additional target networks — one for the actor and one for the critic — which are utilized exclusively for computing the target Q-values. These target networks remain distinct from the primary actor and critic networks. While the primary networks are updated at every training step, their weights are used to softly update the corresponding target networks over time.

Although the theoretical foundations and implementation strategies for DDPG and Q-learning are well-established in the literature, their application to real-world traffic optimization scenarios remains underexplored. The novelty of the present study lies in the integration of these algorithms with a high-fidelity microscopic simulation environment—specifically, the Aimsun platform. Moreover, the study introduces new performance evaluation metrics, such as the *speed_score*, which more accurately captures the influence of traffic control on overall traffic flow dynamics.

The research also highlights the advantages of DDPG over Q-learning in environments characterized by continuous action spaces — an aspect that has not been thoroughly examined within the domain of traffic signal control. As such, this work offers both theoretical contributions regarding the stability and adaptability of reinforcement learning algorithms, and practical insights for their deployment in real-world urban traffic networks.

In conclusion, the findings of this study extend the applicability of reinforcement learning techniques, offering a novel perspective on their integration into complex and dynamic traffic management systems.

6. Input data

Utilizing a simulation environment to assess a deep learning-based traffic management system enables comprehensive monitoring of traffic conditions. However, for such a system to be applicable in real-world scenarios, it is imperative that its input data originate from sources that are commonly available within existing urban transportation infrastructures.

Among the most accessible and widely deployed sources of traffic data are traffic detectors—sensors strategically positioned throughout road networks to monitor vehicular activity. Of these, inductive loop detectors embedded beneath the road surface are the most prevalent. These detectors generate real-time data as vehicles traverse them, typically reporting the following key metrics:

- *Vehicle count*: The number of vehicles detected during a specified sampling interval.
- *Average speed*: The mean velocity of vehicles over the sampling period.
- *Occupancy*: The proportion of time during which the detection area is occupied by a vehicle, which serves as a valuable indicator of congestion levels.

To ensure the feasibility of implementing the model in practical settings, its input parameters are constrained to these standard outputs provided by traffic detectors: vehicle count, average speed, and occupancy. Furthermore, the model incorporates a detailed representation of the transport network, including road geometry, intersections, and their interconnections.

This constraint guarantees the model's compatibility with real-world systems, thereby facilitating a smooth transition from simulated environments to actual deployment in urban traffic infrastructures. In accordance with this design principle—limiting data inputs to those available under real-world conditions—a traffic data summary is constructed using vehicle count, average speed, and occupancy metrics.

Consequently, a performance metric referred to as the speed score (*speed_score*) is introduced. For a given detector i, it is calculated as follows:

$$speed_{scorei} = min\left(\frac{avg_{speedi}}{max_{speedi}}, 1.0\right), \tag{4}$$

Here, avg_{speedi} represents the average speed recorded by motion detector i, while max_{speedi} denotes the designated speed limit on the road segment where detector i is situated. Therefore, the resulting speed estimate is bounded within the interval [0, 1]. This normalized metric is the foundation for constructing both the environmental state representation and the reward function within the reinforcement learning framework.

The experimental setup employs a microscopic traffic simulator that advances the simulation in discrete time steps. At each of these steps, the simulator updates the state variables—such as position, velocity, and acceleration—for all vehicles, based on underlying system dynamics. By

default, each simulation step corresponds to a time increment of 0.75 seconds, a setting that remains unchanged throughout this study.

However, this interval is insufficient to capture meaningful fluctuations in traffic flow, as detected by roadside sensors. Thus, a longer temporal window is introduced to aggregate traffic data. This aggregation period is referred to as an episode step, or simply as a "step" when unambiguous.

The process unfolds as follows:

1. Traffic data is collected at each simulation step but is aggregated over each episode step.
2. The aggregated data is subsequently used as input for the DDPG algorithm.
3. To compute a representative score over time, the speed scores from individual detectors are combined using a weighted average, where the weights are proportional to the vehicle counts observed at each detector.
4. The traffic signal timings produced by the DDPG model are applied to the traffic network during the next episode step.

The optimal duration for an episode step was identified via grid search, with the best-performing interval determined to be 120 seconds.

The speed-based metric described in Equation (4) is utilized to encode the environment's state vector. This choice is justified not only by its ability to capture the degree of congestion across the network but also by its incorporation of each road's maximum allowable speed. As a result, the environmental state is represented as a vector in which each component corresponds to a specific detector and is computed as specified in Equation (5).

$$state_i = speed_{scorei}, \tag{5}$$

where $state_i$ is the state vector and $speed_{scorei}$ — is the speed score.

The rationale for employing a speed-based indicator lies in its capacity to reflect overall traffic flow efficiency: a higher value of this indicator implies that vehicles are traveling at speeds closer to the maximum allowable limit on the given road segment, thereby indicating smoother traffic flow conditions.

In real-world traffic systems, regulatory mechanisms such as traffic lights and temporary signage are typically employed to manage vehicle movements. To maintain a focused and controlled simulation environment, this study exclusively considers traffic light control, thereby avoiding the complexities and potential instabilities associated with the direct manipulation of individual signal states. Instead, signal coordination is achieved by alternating predefined phases at intersections — for example, granting a green signal to one direction while the perpendicular direction remains red.

Rather than adjusting the total signal cycle time, the control strategy modifies only the relative durations of individual phases, thereby preserving temporal synchronization between adjacent intersections. This approach facilitates more coherent traffic patterns across the network. Phase durations are normalized using the softmax function — also known as the normalized exponential function — which ensures that the sum of all phase durations remains constant. Moreover, a scaling factor is applied to constrain the adjustments to 80% of the total cycle, preserving a minimum phase duration for operational stability.

The reinforcement learning algorithm receives feedback via a reward signal, which quantifies the effectiveness of previous actions. While traditional approaches frequently employ

travel-time-related metrics such as queue lengths or delays, the present method capitalizes on data that is readily obtainable from real-world traffic detectors — namely, vehicle count and average speed. The chosen reward metric is the speed estimate; however, to isolate the agent's contribution, a baseline is established using the speed estimate derived from a simulation conducted in the absence of the reinforcement learning agent.

The reward is then computed as the difference between the observed speed estimate and this baseline, weighted by the number of vehicles at each detector to emphasize high-traffic zones. To ensure numerical stability and tractability during training, the resulting value is further scaled by a constant factor α, as defined in Equation (6).

$$reward_i = \alpha \cdot count_i \cdot \left(speed_{scorei} - baseline_i\right) = ¿\alpha \cdot count_i \cdot \left[min\left(\frac{avg_{speedi}}{max_{speedi}}, 1.(\right.\right. \quad (6)$$

,

where $reward_i$ — reward, $baseline_i$ — baseline, $count_i$ — number of vehicles.

In order to normalize the reward values within the range [-1, 1], one potential strategy considered was dividing the reward by the total number of vehicles detected across all sensors. However, this approach was ultimately dismissed, as it introduced inconsistency in reward magnitudes across different simulation stages—specifically, lower vehicle counts at a given time step would artificially inflate reward values, thereby distorting performance evaluations over time.

Instead, a fixed scaling factor α = 1/50 was empirically selected. This factor effectively maintains the scaled reward values within a manageable range, typically close to 1.0, thereby supporting stable gradient updates during the training process. Crucially, this method preserves the relative magnitudes of the original rewards, avoiding the loss of informative variability that would result from more aggressive normalization techniques.

Each traffic detector computes its reward independently at every simulation step, without aggregating values across detectors. These individual rewards are then directly input into the DDPG optimization algorithm. Given the stochastic characteristics of the microscopic traffic simulator, the simulation results are sensitive to the choice of random seed. To mitigate this variability and ensure consistent baseline comparisons, simulations used for benchmarking are conducted with fixed, predefined seed values.

7. Experiment setup

To assess the performance of the deep neural policy (NP) algorithm, a series of simulations were conducted using traffic networks of progressively increasing complexity. The Deep Deterministic Policy Gradient (DDPG) algorithm governs all traffic light phases across the network by leveraging input data from all detectors. For comparative purposes, two alternative agents were employed: a Q-learning agent and a baseline random agent. The Q-learning agent independently controls each traffic light phase using input from local detectors and operates in discrete state and action spaces. In contrast, the random agent assigns arbitrary phase durations drawn from a uniform distribution.

The Q-learning implementation utilizes mosaic coding to discretize continuous state variables into four distinct intervals. It adjusts phase durations based on a fixed set of predefined

ratios (e.g., 0.2, 0.5, 1.0), ensuring that the total cycle duration remains constant through proportional phase adjustments. This architecture adopts a multi-agent configuration, in which agents share some common input data but operate independently when selecting actions.

The random agent assigns phase durations randomly within the interval [0, 1], subsequently scaling the values to maintain consistent cycle lengths. Given the inherent stochasticity of the microscopic traffic simulator, all experiments were conducted using randomized seed values to prevent overfitting and to ensure the robustness of the results. The outcomes were averaged over multiple simulation runs and reported using statistical aggregates (e.g., mean, maximum, minimum) to enable consistent comparative analysis.

The baseline scenario, depicted in Figure 1, features a simplified traffic network consisting of a single intersection formed by two bidirectional two-lane roads. Vehicles may travel straight or turn right at the intersection, while left turns are prohibited, thereby reducing the complexity of both traffic dynamics and signal control logic. The intersection is equipped with eight detectors —one positioned before and one after the intersection on each road.

The traffic signal at the intersection operates on a two-phase cycle. Phase 1 enables movement along the horizontal road, while Phase 2 permits vertical traffic flow. Phase 1 has a duration of 15 seconds, and Phase 2 lasts for 70 seconds, with a 5-second interphase period between them. These deliberately unbalanced phase durations are designed to induce vehicle accumulation along the horizontal road, thereby creating an opportunity for the learning algorithm to demonstrate improvement by adjusting phase lengths. Each simulation run spans a duration of one hour, with a constant vehicle demand of 150 vehicles per centroid pair.

8. Results

To evaluate the performance of the DDPG algorithm in comparison with conventional Q-learning and a random timing baseline on a specified test network, the primary evaluation metric is the average reward per episode. As defined in Equation (5), the reward is represented as a vector, with each element corresponding to a distinct traffic detector within the network. Consequently, the average reward is computed as the mean value across all detector-specific rewards within an episode. For consistency and comparability, the evaluation is based on the best-performing experimental trial, which is defined as the trial that achieves the highest average reward per episode throughout the simulation.

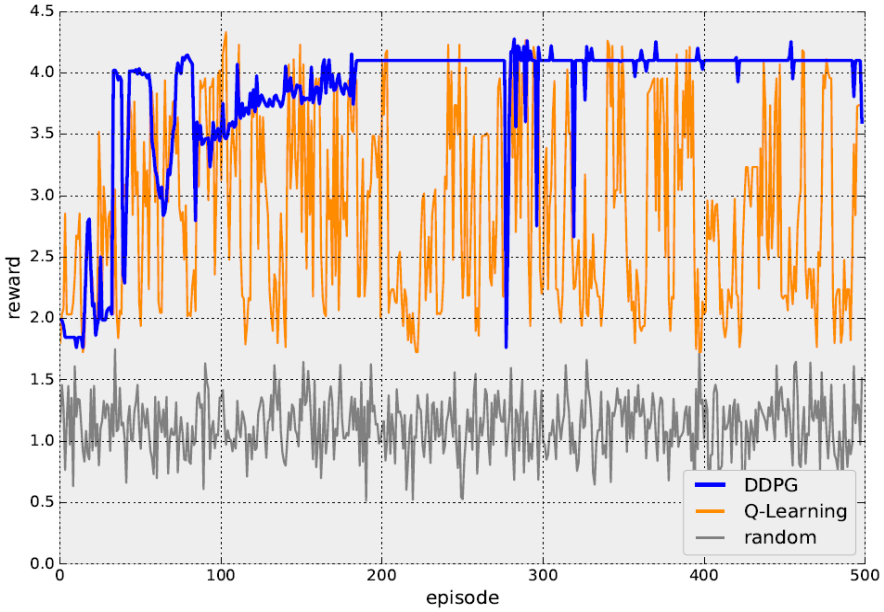

Figure 2: Comparison of algorithm performance in the network.

Figure 2 presents a comparative analysis of performance across different algorithms. Both the DDPG method and classical Q-learning attain comparable levels of average reward. However, notable differences emerge in their convergence behaviors. While Q-learning exhibits considerable instability and fluctuation in reward values over time, the DDPG approach demonstrates significantly greater stability, maintaining consistent performance once peak reward levels are achieved.

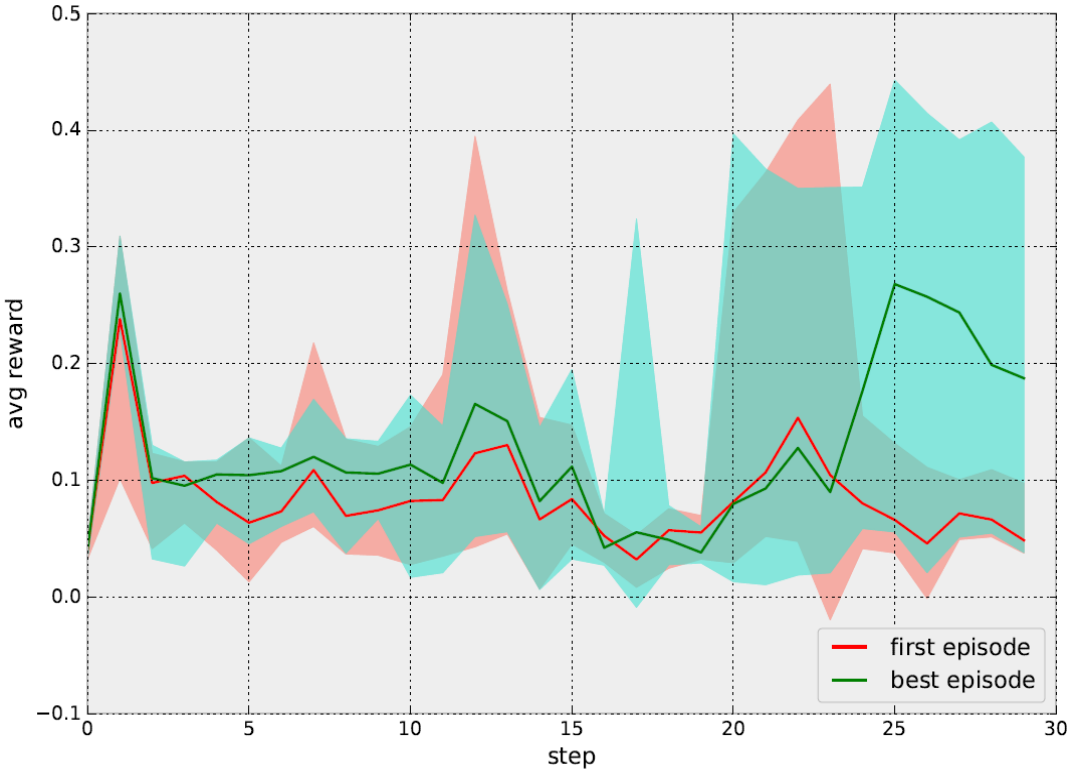

Figure 3: Evolution of the DDPG algorithm in the network in an episode.

Further insights into the algorithm's behavior can be gained by examining the performance within individual episodes. Figure 3 illustrates the results of the first and the most successful episodes of the DDPG algorithm. Performance is represented as the average reward per step, accompanied by the corresponding minimum and maximum ranges. These statistics (mean, minimum, and maximum) are computed across all trials conducted within a single experimental run. The results indicate that, while improvements over the baseline are not uniformly maintained throughout the episode, they remain evident up to its conclusion.

A similar behavioral pattern is observed in the performance of the Q-learning algorithm, as depicted in Figure 4.

It is also essential to acknowledge several limitations of the present study:

- **Model Simplicity:** The simulation is based on a simplified traffic network consisting of a single intersection, which does not adequately capture the complexity of real-world urban transportation systems. In practice, urban environments consist of numerous interconnected intersections, multidirectional traffic flows, and more dynamic traffic behavior.

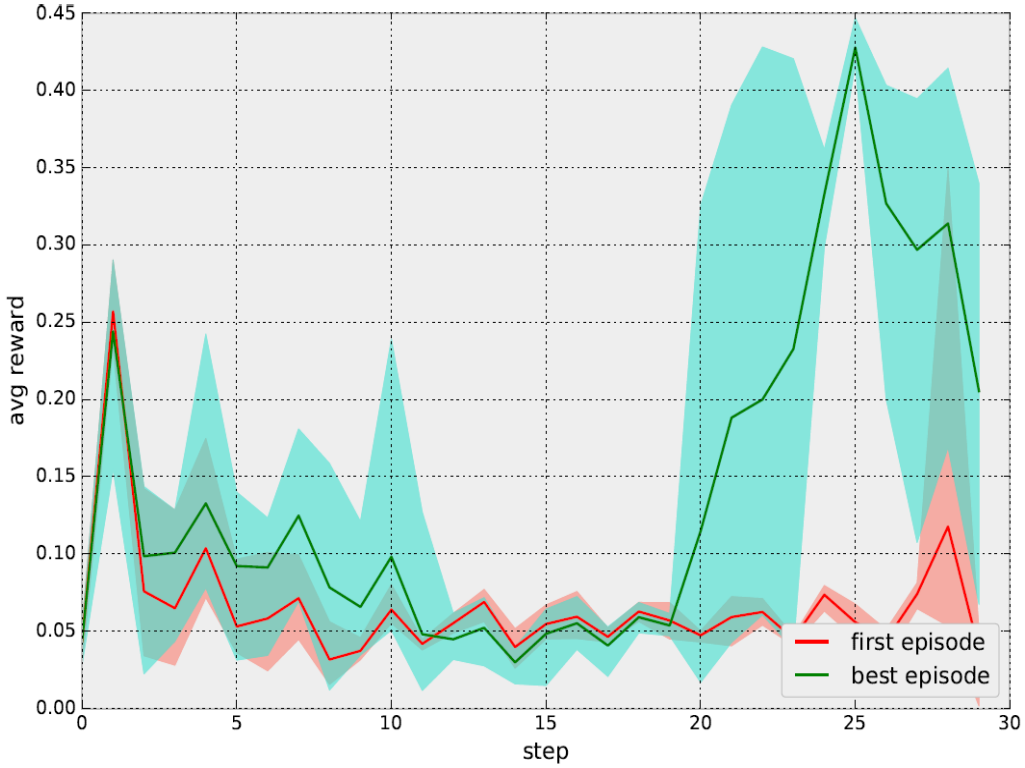

Figure 4: Evolution of the Q-learning algorithm in the network in an episode.

- **Parameter Sensitivity:** The effectiveness of the algorithm is highly dependent on the choice of hyperparameters, such as the discount factor and neural network configurations. Suboptimal parameter selection may negatively impact both the algorithm's performance and its stability.
- **Data Limitations:** The simulation assumes ideal conditions, including precise sensor measurements and the absence of unpredictable variables such as weather fluctuations, traffic incidents, or driver behavior. These real-world factors can substantially influence the effectiveness of the proposed algorithms.

9. Conclusion

The primary objective of this study was to experimentally assess the effectiveness of the DDPG algorithm in optimizing traffic flow management within urban settings. Based on the conducted simulations, the following key findings were obtained:

- The DDPG algorithm exhibited high efficiency and robustness within a simplified transportation network, achieving consistent convergence in contrast to Q-learning, which demonstrated comparatively lower stability.
- Experimental results indicated that DDPG consistently increased the average reward per episode toward the latter stages, suggesting its capacity to adapt to evolving traffic conditions.

- The introduced evaluation metric — the speed score — provided a more nuanced assessment of the algorithm's influence on traffic dynamics and proved to be a valuable tool for training and performance evaluation.

Despite the successful attainment of the study's objective, several limitations must be acknowledged. These include the simplified nature of the model and the absence of validation using real-world data, both of which warrant further investigation.

Future research should focus on extending the model to encompass more complex transportation networks and on validating the approach using data from actual urban environments. Such advancements would enable a more comprehensive evaluation of the DDPG algorithm's applicability to real-world traffic management systems.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] E. van der Pol. Deep reinforcement learning for coordination in traffic light control, August 2016. URL: https://www.researchgate.net/publication/315810688_Deep_Reinforcement_Learning_for_Coordination_in_Traffic_Light_Control_MSc_thesis.

[2] S.R. Qasim, H. Mahmood, and F. Shafait, "Rethinking Table Recognition using Graph Neural Networks," in ICDAR, 2019, URL: https://doi.org/10.48550/arXiv.1905.13391.

[3] P. LA and S. Bhatnagar, Reinforcement Learning With Function Approximation for Traffic Signal Control. IEEE Transactions on Intelligent Transportation Systems, 2012, vol. 12, no. 2, pp. 412-421. URL: https://doi.org/10.1109/TITS.2010.2091408.

[4] S. Acharya, K. K. Dash, R. Chaini. Fuzzy Logic: An Advanced Approach to Traffic Control. Learning and Analytics in Intelligent Systems. International Journal of Innovation in the Digital Economy, 2020, vol. 5, no. 1, p. 10. URL: https://dx.doi.org/10.4018/ijide.2014010103

[5] S. Qu, M. Abouheaf, W. Gueaieb, and D. Spinello, "An Adaptive Fuzzy Reinforcement Learning Cooperative Approach for the Autonomous Control of Flock Systems". ICRA, 2021, pp. 8927-8933. URL: https://ieeexplore.ieee.org/document/9561204.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, & M. Riedmiller. Playing Atari with Deep Reinforcement Learning, 2013. URL: https://doi.org/10.48550/arXiv.1312.5602.

[7] M. Ghosh, A. Roy, A. Jha, R. Saha, and S. Ghosh, "Performance analysis of deep Q-learning based agent for playing Atari 2600 games," in 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, Canada, Oct. 2019, pp. 0429–0435. URL: https://ieeexplore.ieee.org/document/8936153.

[8] Vinyals, O., Babuschkin, I., Czarnecki, W.M. et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature 575, 350–354 (2019). URL: https://www.nature.com/articles/s41586-019-1724-z.

[9] D. Kingma, J. Ba. Adam: A method for stochastic optimization. 3rd International Conference for Learning Representations, San Diego, 2015. URL: https://arxiv.org/abs/1412.6980.

[10] R. Sutton, D. A. McAllister, S. Singh, Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. NIPS'99: Proceedings of the 13th International Conference on Neural Information Processing Systems, 1999, pp. 1057 – 1063. URL: https://dl.acm.org/doi/10.5555/3009657.3009806.

[11] D. Chernyshev, S. Dolhopolov, T. Honcharenko, V. Sapaiev and M. Delembovskyi. "Digital Object Detection of Construction Site Based on Building Information Modeling and Artificial Intelligence Systems", CEUR Workshop Proceedings, 2022, 3039 pp. 267–279. URL: https://doi.org/10.48550/arXiv.1509.02971.

[12] Timothy R., Yanzhi W., A survey for deep reinforcement learning in Markovian cyber-physical systems: Common problems and solutions, Neural Networks, Volume 153, 2022, pp. 13-36. URL: https://doi.org/10.1016/j.neunet.2022.05.013.

[13] Aimsun Next user manual, version 24.0.1. URL: https://docs.aimsun.com/next/24.0.1/.

[14] V. Levytskyi, Kruk, O. Lopuha, D. Sereda, V. Sapaiev, O. Matsiievskyi. Use of Deep Learning Methodologies in Combination with Reinforcement Techniques within Autonomous Mobile Cyber-physical Systems, 2024 IEEE. URL: http://dx.doi.org/10.1109/SIST61555.2024.10629589.